2020

A BLOCKCHAIN-BASED AUDITABLE AND SECURE VOTING SYSTEM

Madhukara Kekulandara
University of Rhode Island, mkekulandara@uri.edu

A BLOCKCHAIN-BASED AUDITABLE AND SECURE VOTING SYSTEM

BY

MADHUKARA KEKULANDARA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

UNIVERSITY OF RHODE ISLAND

2020

MASTER OF SCIENCE THESIS

OF

MADHUKARA KEKULANDARA

APPROVED:

Thesis Committee:

Major Professor Edmund Lamagna

Lutz Hamel

Gavino Puggioni

Brenton DeBoef
DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2020

ABSTRACT

Improving electronic voting systems to provide election security and integrity while controlling cost has been an area of active research for decades. As a result, many technological improvements are incorporated into the voting systems used today. The introduction of technology, however, has not been without issues and has raised new concerns. One is the possibility of inaccurate election outcomes due to technical failures of the equipment. Another is the problem of election security and the possibility of malicious alteration of election results. Yet another concern is the capability to conduct post-election audits to validate and provide confidence in election results.

The research reported here applies the features of blockchains and zero-knowledge protocols to improve the security, integrity, and transparency of electronic voting systems. This study proposes a new voting algorithm that can be used as an extension to the existing voting systems to provide evidence about the accuracy of an election. A prototype system is developed and implemented, and the system's security and auditing features are tested. The Rhode Island voting system is used as a case study in this research. The proposed algorithm is compatible with current election technology and addresses many major concerns about present voting systems.

ACKNOWLEDGMENTS

I would like to express my sincerest gratitude towards my major professor and thesis advisor, Dr. Edmund Lamagna for his willingness to spend long hours working with me to design this voting system. His outstanding suggestions throughout the process provided great support to achieve the expected goals of my research.

I would also like to thank my committee members, Dr. Lutz Hamel and Dr. Gavino Puggioni for offering their time and valuable suggestions. I also like to thank Dr. Gretchen A. Macht for her valuable comments and also for chairing my defense.

I feel a deep sense of gratitude towards my father, sisters, and all my friends for their support, encouragement and love. The confidence brought to me by their words made this research possible.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

CHAPTER 1

Introduction

Electronic voting systems have been the subject of active research for decades. The goal of such work has been to minimize the cost of conducting an election while maintaining the security and integrity of the election, as well as voter privacy. These studies have contributed many improvements to the voting systems we use today. Optical ballot scanners, paperless voting systems, encrypted voting systems and internet voting systems are some important outcomes of such work.

Many states, including Rhode Island, have adopted and are using these technologies in elections. Rhode Island decided to move from mechanical lever machines to optical scan precinct count voting systems in 1997 [1]. The first election to be conducted over the Internet in the US was the 1996 Reform Party Presidential primary, in which Internet voting was offered, along with vote-by-mail and vote-by-phone, as an option to party members who did not attend the party convention [2]. Georgia became the first state to implement the use of direct recording electronic voting machines on a statewide basis, deploying the DREs at the same time in every county [3].

As a result of widespread adoption of electronic voting systems, U.S. elections currently rely heavily on the quality of the technology used [4]. In the year 2000, a controversial recount occurred during the presidential election in the state of Florida [5]. During the November 2004 general election in Carteret County, North Carolina, electronic voting machines lost 4,438 votes [6]. These and many other incidents involving close races—including a 1978 election for the Rhode Island Senate, one in 1996 for the South Dakota House of Representatives, and a 1988 Massachusetts Senate Democratic primary [7]—have brought the integrity of

existing voting technologies into question. One of main concerns is the possibility of inaccurate election outcomes occurring as a result of technical failures of election equipment. Technical failures and weaknesses in the security also enabled unauthorized modification of the election process and the final tally. Lack of transparency and the inability to conduct post-election audits cause people to lose their trust in the election process.

As a result of such concerns, a list of compliance suggestions for electronic voting, called the Voluntary Voting System Guidelines (VVSG), was issued by the U.S. Election Assistance Commission (EAC) in 2005. Many states adopted these regulations to overcome some of the weaknesses in their voting systems. For example, Nevada became the first state to mandate that all electronic voting machines used in federal elections be equipped with printers that produce a voter-verified paper audit trail [8]. The California Secretary of State, Kevin Shelley, decertified all touchscreen electronic voting machines in the state and banned their use in four counties until significant improvements were made to the security of the systems [9]. Maryland Governor Robert L. Ehrlich, Jr., publicly urged voters to vote by absentee paper ballot instead of using the state's electronic voting machines in the November 2006 General Election after problems with the machines emerged during Maryland's primary that year [10]. Unfortunately, insufficient changes have been made to improve the quality of the existing voting systems used by most states. [4]

Security breaches of existing voting systems fall into two categories of the voting process: those involving voting machines at the precinct level, and those involving the centralized servers where the results are aggregated. Machines used by voters have been viewed as flawed, due mainly to security concerns. Anyone with physical access to a voting machine can sabotage it, thereby affecting all

votes cast [1]. In 2011, a group of computer science and security experts on the Vulnerability Assessment Team at Argonne National Laboratory in Illinois managed to hack a commonly used electronic voting machine using a remote control that cost less than $11 [11]. This issue becomes even more critical at the backend of an election system, when the results of voting machines are forwarded to central processing centers, where servers that hold the election results are even more vulnerable to cyber-attacks. In December 2005, Black Box Voting, Inc., set up a demonstration in Leon County, Florida, where computer security experts Harri Hursti and Herbert Thompson were able to hack into the central vote tabulator of an electronic voting system and change the outcome of a mock election without leaving any trace of their actions [12]. This is one of the key areas where blockchain technology can be beneficial to a voting system.

Another major concern about current voting systems is their capability to conduct post-election audits of the election results. "A voting system that may produce accurate results, but provides no way to know whether it did, is inadequate. It provides far too many ways for resourceful adversaries to undermine public confidence in election integrity" [13]. To address this concern, a strategy was introduced by Philip B. Stark and David A. Wagner in 2012 to conduct evidence-based elections [14]. This strategy involves three main points: use paper ballots, protect them, and check them. More specifically:

1. Voters must vote by marking paper ballots - either manually or using ballot marking devices. In either case, there should be a convenient and accessible way for voters to verify their ballots and, when necessary, to mark a replacement ballot before officially casting their vote.

2. Voted paper ballots must be carefully stored and managed to ensure that no ballots are added, removed or altered, and procedures should be established

to provide strong evidence of proper ballot management.

3. Voted ballots also must be checked in robust post-election vote tabulation audits. This procedure should involve audit judges manually reviewing a random sample of cast ballots and comparing them to the reported initial counts before the election results are finalized. These audits should be risk-limiting audits (RLAs), which are very likely to correct any election outcome that is incorrect due to a mistabulation of votes. In very close elections, a full manual count may be required. [13]

The use of paper ballots is strongly recommended as it leaves an auditable trail. Blockchains, however, can provide viable paperless audit trails as a substitute for this recommendation. Blockchains are one of the most secure data structures to hold sensitive information, and incorporate sufficient capabilities to conduct audits of the information stored in the chain.

The blockchain technology was invented by a person (or group of people) known as Satoshi Nakamoto in 2008 [15]. Its most widely known use to date is in maintaining public transaction ledgers for cryptocurrencies. It is "an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way" [16]. Blockchains have many features to create resistance to alteration of the data stored in the blocks. Once recorded, the data in any given block cannot be altered retroactively without alteration of all subsequent blocks(See Figure 1), which requires consensus of the network majority.

A blockchain possess four main features:

- The ledger exists in many different locations. Hence it is impossible to tamper with the content of a blockchain by changing the contents at one location.

- There is distributed control over who can append new transactions to the

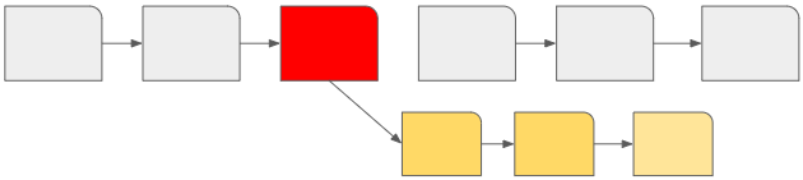

Figure 1. Alteration of a blockchain

ledger.

- Any proposed "new block" to the ledger must reference the previous state of the ledger, creating an immutable chain.

- A majority of the network nodes must reach a consensus before a proposed new block of entries becomes a permanent part of the ledger [17].

To date, the principal use of blockchains has been in cryptocurrency, most notably Bitcoin [15]. However, blockchains are increasingly being used for a number of other applications because of their inherent resistance to the modification of a transaction, block, or the entire distributed ledger [18]. Mediachain is a peer-to-peer, decentralized database for sharing information across applications and organizations [19]. Propy is a Silicon Valley-based Cryptocurrency Company working towards modernizing the real estate industry through the use of Blockchain technology [20].

Blockchain technology provides a potential solution to many security problems associated with voting systems:

1. Inherent resistance to modification can be used as a shield against any attempt at tampering with the recorded votes.

2. Since the ledger exists in many different locations, cyberattacks on a single server will not cause the entire system to fail.

3. A consensus is required before new block entries become permanent, avoiding the addition of illegal blocks (votes) to the chain.

4. Blockchains also provide a capability to conduct election audits even when no paper trail is available.

Introducing new technologies into a system that already suffers from technological failures might not be a feasible solution. A new voting algorithm purely based on a blockchain network that uses coins as votes will create new security challenges rather than solving the existing ones [21]. Despite that, we can still use some of the key features in blockchains like proof of work, consensus mechanism, and hash links to create a partially decentralized chain of ballots that can provide proofs to the results posted by the existing voting system. These proofs can be used as evidence to validate the elections or to identify any attempt of malicious activity during an election.

This research addresses the use of features in blockchains and zero-knowledge protocols to improve the security, integrity and the transparency of electronic voting systems. The goal is to design an extension to the existing election system that will improve the security and integrity of current ones, while at the same time facilitating the auditing of election results. The Rhode Island voting system is used as the case study because of our familiarity with it and our ability to ask questions to local voting authorities as they arise. Another of our goals is to introduce a minimum of changes to existing voting systems and that are compatible with the current election process.

List of References

[1] "State of rhode island general assembly. rhode island general laws, title17 - elections - chapter 17-19 conduct of election and voting equipment,and supplies," http://webserver.rilin.state.ri.us/Statutes/TITLE17/17-19/17-19-2.1.HTM, (Accessed on 10/26/2020).

[2] L. F. Cranor, *In Search of the Perfect Voting Technology: No Easy Answers*. Boston, MA: Springer US, 2003, pp. 17–30. [Online]. Available: https://doi.org/10.1007/978-1-4615-0239-5_2

[3] J. Cathy Cox, "Georgia's unique model for election reform," https://votingmachines.procon.org/historical-timeline/, Nov 2002, (Accessed on 10/26/2020).

[4] S. I. Mello, *A detailed forensic analysis and recommendations for Rhode Island's present and future voting systems*. University of Rhode Island, 2011.

[5] "2000 united states presidential election in florida - wikipedia," https://en.wikipedia.org/wiki/2000_United_States_presidential_election_in_Florida, (Accessed on 10/26/2020).

[6] E. Theisen, "Myth breakers: Facts about electronic elections," http://www.votersunite.org/mb2.pdf, (Accessed on 10/26/2020).

[7] "List of close election results - wikipedia," https://en.wikipedia.org/wiki/List_of_close_election_results, (Accessed on 10/26/2020).

[8] D. A. Heller, "Certification of voter-verified paper audit trail printer completed," https://votingmachines.procon.org/historical-timeline/, July 2004, (Accessed on 10/26/2020).

[9] J. Kevin Shelley, "California secretary of state news release," https://votingmachines.procon.org/historical-timeline/, (Accessed on 10/26/2020).

[10] C. Davenport, "Democrats blast ehrlich's absentee-voting initiative," https://votingmachines.procon.org/historical-timeline/, (Accessed on 10/26/2020).

[11] B. FRIEDMAN, "Diebold voting machines can be hacked by remote control," https://www.salon.com/2011/09/27/votinghack/, (Accessed on 10/26/2020).

[12] M. L. Songini, "Expert calls for increased e-voting security," https://www.computerworld.com/article/2560901/expert-calls-for-increased-e-voting-security.html, (Accessed on 10/26/2020).

[13] "Pilot implementation study of risk-limiting audit methods in the state of rhode island," https://verifiedvoting.org/wp-content/uploads/2020/07/RI-RLA-Report-2020.pdf, August 2019, (Accessed on 10/26/2020).

[14] P. B. Stark and D. Wagner, "Evidence-based elections," *IEEE Security & Privacy*, vol. 10, no. 5, pp. 33–41, 2012.

[15] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[16] M. Iansiti and K. Lakhani, "The truth about blockchain:," *Harvard business review*, vol. 95, pp. 118–127, 01 2017.

[17] F. P. Hjalmarsson, G. K. Hreioarsson, M. Hamdaqa, and G. Hjalmtysson, "Blockchain-based e-voting system," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. Los Alamitos, CA, USA: IEEE Computer Society, jul 2018, pp. 983–986. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CLOUD.2018.00151

[18] R. Ganji, "Electronic voting system using blockchain," https://education.dellemc.com/content/dam/dell-emc/documents/en-us/2018KS_Ganji-Electronic_Voting_System_using_Blockchain.pdf, (Accessed on 10/26/2020).

[19] "Mediachain : Documentation," http://docs.mediachain.io/, (Accessed on 10/26/2020).

[20] "Propy wiki," https://everipedia.org/wiki/lang_en/propy, (Accessed on 10/26/2020).

[21] S. Park, M. Specter, N. Narula, and R. L. Rivest, "Going from bad to worse: from internet voting to blockchain voting."

CHAPTER 2

Prior Related Work

In this chapter, the most prominent voting systems that influenced our work are surveyed. The ballot structure, the voting process, and auditing capabilities of each system are described briefly. This chapter also discusses the auditing requirements necessary for an election in Rhode Island and presents a description of the machines used in the state. At the end of the chapter, a brief introduction to all the technologies used in our work is presented.

2.1 Voting Systems

2.1.1 Aperio

Aperio is a paper-based voting system that allows the creation of verifiable audit trails without involving any cryptographic methods. It is an 'end-to-end' integrity verification mechanism that can be used in secret paper-ballot environments without the use of sophisticated election machines.

Aperio uses a randomized candidate order on each ballot paper. This allows the generation of a set of paper receipts with the voter's mark in its proper location, but without exposing candidate names. As a result, it provides auditability while maintaining voter privacy. This system was first presented at the WOTE2008 conference by Aleks Essex et. al. of the University of Ottawa as a way to conduct high integrity elections in countries that have limited access to technology [1].

Aperio uses a stack of ballot papers instead of a single paper ballot. This stack is referred to as the "ballot assembly". It consists of four (or more) sheets of paper separated by carbon paper. In this stack, the first sheet is the ballot itself with the candidate names in randomized order. This sheet includes ovals where votes are marked. On the second sheet, a serial number is printed along with the

ovals in the same position as the first sheet, but there are no candidate names. The second sheet is a receipt that a voter retains and can use to verify his/her vote. The last two sheets are audit sheets containing commitment reference numbers that are used during the audit process. Like the second sheet, they include the marked ovals without the candidate names. (See Figure 2)

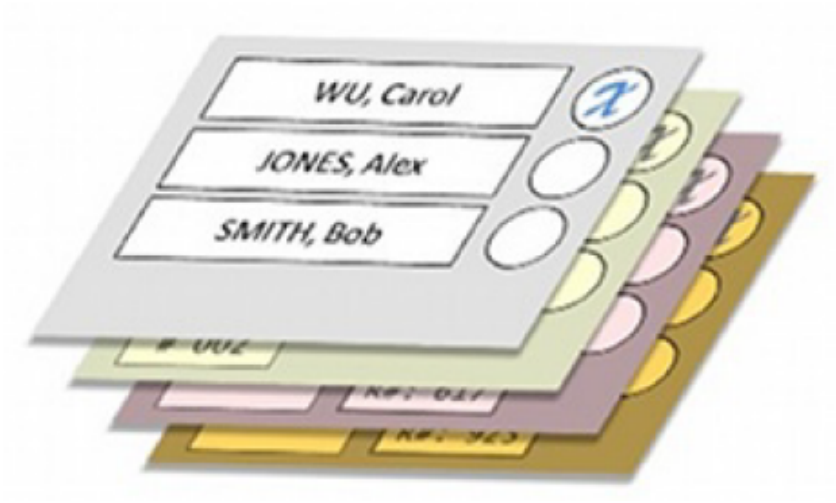

Figure 2. Aperio ballot assembly

There is no limit to the number of audit sheets in the stack. There can be as many audit sheets as desired, one for each group of auditors. When the top ballot is marked, the mark carries through all the ballot sheets because of the carbon paper.

Before election day, two commitment lists are created by the election authority. The ballot commitment list holds information concerning the candidate order for each ballot assembly. The receipt commitment list holds a serial number associated with each ballot assembly. These lists are exposed to the public depending on the type of audits conducted after the election. To conduct a secure audit, only one of these lists is ever revealed. The other is destroyed during the auditing process.

On election day, a voter marks choices on the top ballot and then separates

the ballot assembly. The top sheet goes into a ballot box, the voter retains the second sheet as a receipt, and the audit sheets go into corresponding audit boxes.

Aperio is capable of conducting three types of audits: receipt audits, tally audits, and ballot audit. These are discussed later in this chapter.

2.1.2 Scantegrity II

Scantegrity II is an end-to-end cryptographic voting system created for use with optical scan voting technology. The basic principle of this system is the use of secret codes, called confirmation codes, that are printed in invisible ink on the ballot. Voters reveal an invisible code by marking the intended oval on the ballot. After casting a vote, a voter can check this code on a bulletin board to be confident his vote was counted and included in the final tally. Scantegrity was originally proposed by David Chaum in 2007 [2]. Scantegrity II, introduced in 2009, became somewhat of a success and was used in a governmental election held in Tacoma Park, Maryland [3].

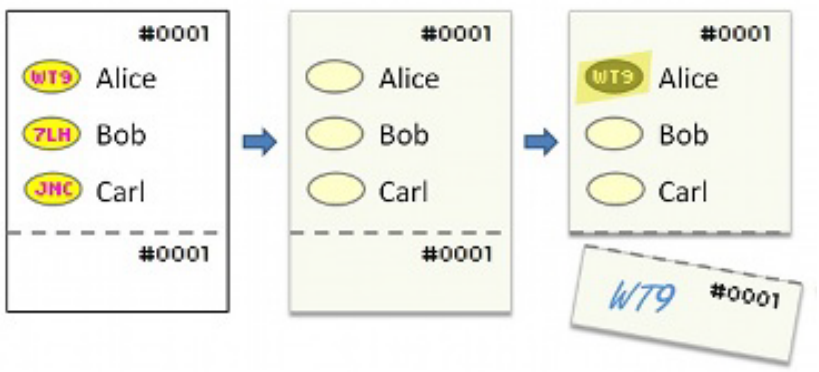

Figure 3. Scantegrity II ballot structure

As shown in Figure 3, a voter wishing to verify his vote can write down the confirmation code appearing in the oval on the receipt and take it home. Unlike Aperio, in Scantegrity II candidate names appear in the same order on every ballot, but there is a unique serial number on every ballot.

The serial number of the ballot and all the secret codes on it are stored in a table referred to as the "ballot table" (Figure 4: Table P). Since this table exposes the relationship between the confirmation code and a candidate, it must never be published. As a result, three commitment tables are created for use in audits: The "permuted ballot table", the "shuffle table", and the "result table".

- The permuted ballot table holds the confirmation codes for each ballot without the candidate's name. To maintain secrecy, the codes are permuted to change the order they appear on the ballot. (Figure 4: Table Q)
- The shuffle table holds details of the confirmation codes printed in invisible ink on each ballot. It also holds a pointer to the result table mapping each candidate's vote to the ballot serial number from which the vote originates. (Figure 4: Table R)
- The Result table holds the final tally of the election. (Figure 4: Table S)

Ballot ID	Alice	Bob	Carl
0001	WT9	7LH	JNC
0002	KMT	TC3	J3K
0003	CH7	3TW	9JH
0004	WJL	KWK	H7T
0005	M39	LTM	HNN

Table **P**

Ballot ID			
0001	7LH	WT9	JNC
0002	J3K	TC3	KMT
0003	9JH	CH7	3TW
0004	KWK	H7T	WJL
0005	M39	HNN	LTM

Table **Q**

Flag	**Q**-Pointer	**S**-Pointer
	(0005, 1)	(2, 1)
	(0003, 3)	(4, 2)
	(0002, 1)	(4, 3)
	(0001, 3)	(3, 3)
	(0001, 2)	(4, 1)
	(0005, 3)	(3, 2)
	(0004, 2)	(5, 3)
	(0003, 1)	(2, 3)
	(0004, 3)	(3, 1)
	(0002, 3)	(1, 1)
	(0001, 1)	(2, 2)
	(0002, 2)	(5, 2)
	(0004, 1)	(1, 2)
	(0003, 2)	(5, 1)
	(0005, 2)	(1, 3)

Table **R**

Alice	Bob	Carl

Table **S**

Figure 4. Scantegrity II commitment tables

Scantegrity II has capabilities for conducting receipt audits, ballot audits, pre-election cut and choose audits, tally audits, and post-election randomized partial checking (RPC) audits.

2.1.3 Helios

Helios is an open-source voting system for strictly online elections. This system is based on public-key cryptography and has an auditing capability. A primary goal of this project was to create a platform enabling anyone to set up and conduct a completely online election. The Helios system was created in 2008 by Ben Adida at MIT [4].

Internet voting is considered by most as an insecure way to conduct elections due to security and privacy vulnerabilities, as well as the lack of a paper ballot trail. As a result, Adida does not endorse Helios for elections that involve high stakes. He suggests this system for schools and clubs conducting low-stakes online elections where there is little or consequence of a cyber-attack.

The Helios system has been successfully used on many occasions. In March 2009 it was deployed in the election of the President of Université Catholique de Louvain in Louvain-la-Neuve, Belgium [4]. It was also used to run the Princeton undergraduate student government election in October 2009.

Since Helios is designed for online elections, the ballots are created as virtual web forms. Cast ballots are encrypted using the El Gamal cryptosystem before being sent back to a server for inclusion in the tally. The private key needed to decrypt a ballot is saved on a trusted workstation. The Helios system is capable of conducting three types of audits: ballot audits, receipt audits, and tally audits [5]. A ballot audit is performed on the virtual ballots by displaying the SHA-1 hash ciphertext of the ballot to the voter.

2.1.4 Prêt à Voter

This system was created by Peter Ryan at Newcastle University in 2004. Like Aperio, it uses a randomized candidate order to provide verifiability while maintaining ballot secrecy [6].

A Prêt à Voter ballot has two halves that can be separated in the middle. The left side is printed with a list of random candidates. And the right-side has boxes where the voter marks his intention with a pen. The right-side also has a 2D bar code containing the information necessary to decrypt the candidate order printed on the left-side. See Figure 5. The key to decrypting the candidate order is encrypted in such a way that no one person alone can decrypt the ballot.

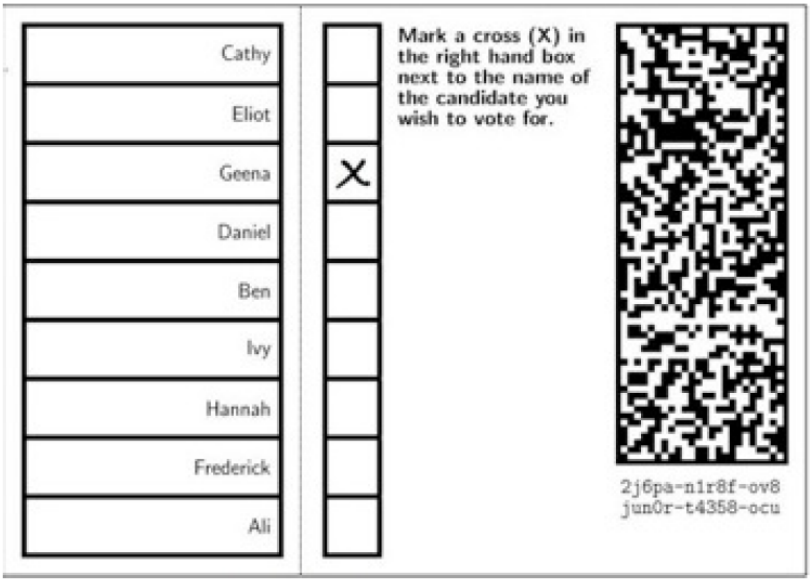

Figure 5. Prêt à Voter ballot

On election day, voters mark their ballots with an "X" using a pen on the right side of the paper, detach the left side from the ballot, and discard it; the right-side is scanned using an optical scanner that records and adds the vote to the tally. Voters leave the polling place with the right side of the ballot as a receipt.

The Prêt à Voter system is capable of conducting four types of audits: ballot

audits, receipt audits, tally audits, and post-election mixnet audits.

2.1.5 WAVERI

WAVERI is an election algorithm based on set theory. The name WAVERI stands for Watch, Audit, Verify Elections for Rhode Island. This algorithm offers a solution that creates verifiable audit trails without the added complexity associated with cryptographic schemes. The algorithm was created in 2011 by Suzanne I. Mello-Stark at the University of Rhode Island [7].

Prior to election day, the algorithm creates a set of unique codes and saves them on a precinct's election system. On election day, the algorithm secretly divides the audit code set into a family of disjoint subsets. One subset is assigned to each candidate in every race. When a vote is cast, an audit code is removed from the selected candidate's subset and placed in the used audit code set. The audit code is printed for the voter to take home for later verification. Since the original candidate subsets are never exposed, the audit code cannot be linked to a specific candidate. Final vote tallies for each candidate are calculated by looking into the unused audit codes in each candidate's subsets.

WAVERI system is capable of conducting four types of audits: receipt audits, tally audits, randomized partial checking audit, and complete set audit

2.2 Election Auditing Methods

2.2.1 Ballot Audits

A ballot audit is used to verify the ballots are printed correctly. A voter or an auditor can conduct a ballot audit on election day. To begin a ballot audit, the interested party asks a poll worker for a blank ballot. The poll worker marks a ballot as an "audit ballot" and hands it to the auditor for verification [7].

2.2.2 Receipt Audits

A receipt audit allows voters and watchdog groups to make sure all the receipts are included in the final tally. After the election day, an auditing group collects ballot receipts from voters and compares the collected receipts with the corresponding ballots. Any missing ballots during the election can be identified using this audit [7].

2.2.3 Tally Audits

A tally audit gives auditors another way to verify the vote tally. There are various methods to perform this audit based on the voting system. The main goal is to provide evidence of the correctness of the final tally.

2.3 Risk Limiting Audits

In October 2017, the Governor of Rhode Island signed into law a groundbreaking election security measure requiring Rhode Island election officials to conduct risk-limiting audits (RLAs) staring with the 2020 presidential primary [8]. According to this law, election officials must conduct an RLA on a random sample of cast ballots determined by statistical modeling instead of auditing a predetermined number of ballots [8].

There are three different approaches to risk-limiting audits.

- Ballot-level comparison: a random sample of cast ballots is manually interpreted, and each manual interpretation is checked against the machine interpretation of the same ballot.

- Ballot polling: a random sample of voted ballots is manually interpreted, and the resulting manual vote counts are checked against the total machine counts to see if they provide strong statistical evidence that the reported outcome is correct. This method is very similar to exit polling.

- Batch level comparison: a random sample of "batches" is selected, and the votes in each batch are counted manually. A batch may consist of all the ballots cast in a precinct, or on a particular voting machine. The counts are compared to the corresponding precinct or machine counts, batch by batch, to determine any discrepancies.

2.4 Voting Machines in Rhode Island

2.4.1 ES&S DS200

The ES&S DS200 is a precinct-based, voter-activated paper ballot counter and tabulator. The DS200 has a 12" LCD touch screen that provides voters with feedback, such as an overvote warning. When the polls close, the ES&S DS200 prints out logs providing election officials with a paper tally of the votes cast. The DS200 captures digitized images of all ballots scanned. Write-in votes and problematic ballot markings can be processed using the digitized images. Consequently, once the ballots are scanned, they need not be handled except in the event of a recount or audit [9]. This system is used as the ballot scanner at all polling places in Rhode Island elections. All ballots are marked by hand or, for accessibility purposes, an AutoMark device.

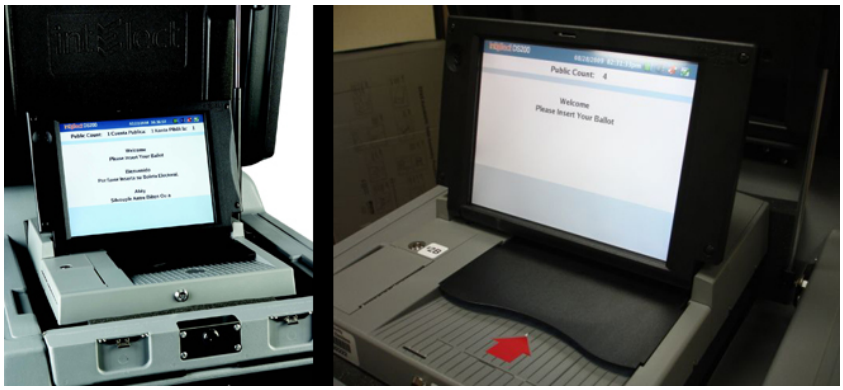

Figure 6. ES&S DS200 voting machine

2.4.2 ES&S DS850

The DS850 is a high-speed, digital scan ballot tabulator designed for use by election officials at a central election facility. The DS850 can scan and count up to 300 ballots per minute. It uses digital cameras and imaging systems to read the front and back of each ballot, evaluate the result, and sort each ballot into an appropriate tray based on the result to maintain continuous scanning and tabulating. Multiple criteria can be used to segregate ballots for review including overvotes, crossover votes, and blank ballots. Depending on the situation, ballots segregated in this fashion may not be counted and may need to be remade by the election inspectors. Rhode Island mainly uses these systems for mail and absentee ballot tabulation [10].

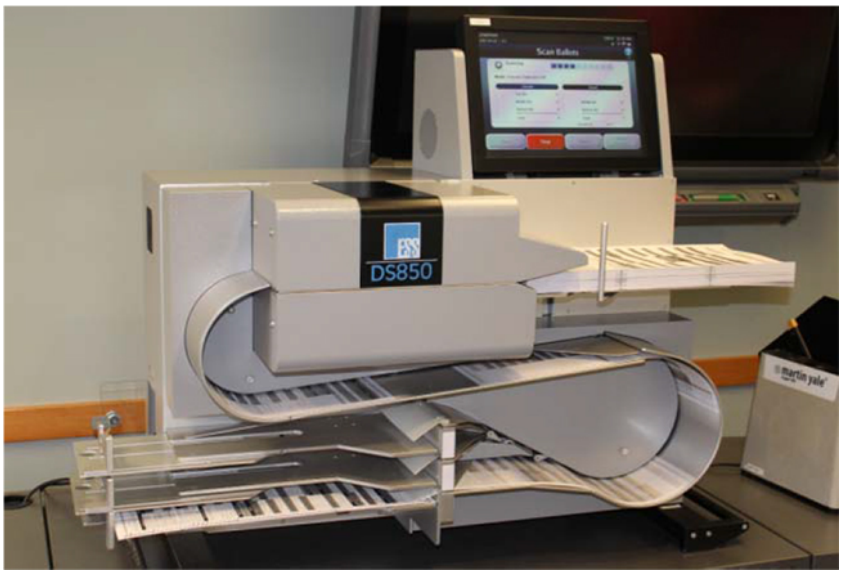

Figure 7. ES&S DS850 voting machine

2.4.3 ES&S AutoMARK

The AutoMARK Voter Assist Terminal (VAT) is an optical scan ballot marker designed for use by people who are unable to mark an optical scan ballot due to physical impairments or language barriers. Originally patented by Eugene Cum-

mings in 2003, ES&S purchased the rights to manufacture and distribute the systems in 2008 [11].

Figure 8. ES&S AutoMARK ballot marking device

2.5 Blockchain Technology

The first-ever blockchain-like protocol was proposed by cryptographer David Chaum in his 1982 dissertation, "Computer Systems Established, Maintained, and Trusted by Mutually Suspicious Groups"[13]. This work was continued in 1991 by Stuart Haber and W. Scott Stornetta into a cryptographically secured chain of blocks [12]. Their goal was implementing a system where document timestamps could not be tampered.

In 2008, the notion of blockchain was conceptualized by a person (or group of people) known as Satoshi Nakamoto [13]. Nakamoto's design used a Hashcash-like method to timestamp blocks without requiring them to be signed by a trusted party [14]. In the following year, Nakamoto introduced a cryptocurrency called Bitcoin, where blockchain technology serves as the basis for implementing a public ledger used to support this digital currency [13].

A blockchain is a system of recording information in a way that makes it difficult or impossible to change, hack, or cheat the system [15]. A blockchain ledger exists in many different locations. Hence it is impossible to tamper with the content of a blockchain by changing the information stored at just one location. There is distributed control over who can append new transactions to the ledger. Any proposed "new block" in the ledger must reference its previous state, creating an immutable chain. A majority of the network nodes must reach a consensus before any proposed new block of entries becomes a permanent part of the ledger.

To date, the principal use of blockchains has been in cryptocurrency, mainly Bitcoin. However, blockchains are increasingly being used in other applications including "smart contracts" [16], financial services, video games, energy trading, supply chains, and domain name registration [17].

2.6 Proof of Work

A proof-of-work (POW) system (or protocol, or function) is a consensus mechanism whose main purpose is to prevent denial-of-service attacks and other abuses such as spam on a computer network. A service requester is required to perform some work, usually equating to computer processing time, in order to receive a requested service. The concept was invented by Cynthia Dwork and Moni Naor in a 1993 journal article [18]. The term "proof of work" was first introduced and formalized in 1999 by Markus Jakobsson and Ari Juels [19].

A key feature of proof-of-work schemes is their asymmetry. The work must be moderately hard, though feasible, for the requester to perform, but easy for the service provider to check. This notion is also known as a CPU cost function, client puzzle, computational puzzle, or CPU pricing function. It is different from a CAPTCHA, which is intended for a human to solve quickly while being difficult for a computer to solve.

2.7 Related Technologies

2.7.1 Cryptographic Hash Functions

A cryptographic hash function, or hashing algorithm, takes a block of data and operates on it in a deterministic fashion to scramble the information and produce a much smaller fixed-size string called a hash value. A good hash function should have the property that it is infeasible for two distinct data blocks to produce the same hash value. These functions were originally invented in the 1950s to detect errors in communications [20].

One of the first hash functions to gain acceptance was MD5, developed by Ron Rivest in 1991 [21]. A pair of strings producing the same value was reported in 2004 and several other collisions have been found since [22]. Consequently, the method is no longer considered strongly collision-resistant and MD5 is not recommended for use in secure applications.

SHA-1 (Secure Hash Algorithm) is a widely used hash function designed by the NSA. Although this method produces a much larger 160-bit hash, collisions were reported in 2005 and 2008 [23].

SHA-2 and SHA256 are newer versions of the SHA-1 hash algorithm. SHA256 produces a 256-bit (32 bytes) hash value that is usually reported as 64-digit hexadecimal number.

2.7.2 Zero-knowledge Proof

Zero-knowledge proof (protocol) is a method by which one party (the prover) can demonstrate to another (the verifier) that they know a value x, without conveying any information apart from the fact that they know the value x [26]. The base of this technique is to use mathematical methods to verify things without sharing or revealing underlying data.

This method was initially introduced in 1989 by Shafi Goldwasser, Silvio Mi-

cali, and Charles Rackoff in their paper "The Knowledge Complexity of Interactive Proof-Systems" [28]. It has been successfully used in many areas including authentication systems, to enforce ethical behavior (according to community standards) among members of an online community, nuclear disarmament, and the blockchain.

List of References

[1] A. Essex, J. Clark, and C. Adams, "Aperio: High integrity elections for developing countries," in *Towards Trustworthy Elections*. Springer, 2010, pp. 388–401.

[2] D. Chaum *et al.*, "The scantegrity system, an introductory whitepaper and example," *Last accessed on January*, vol. 10, p. 2010, 2011.

[3] R. Carback, D. Chaum, J. Clark, J. Conway, A. Essex, P. S. Herrnson, T. Mayberry, S. Popoveniuc, R. L. Rivest, E. Shen, *et al.*, "Scantegrity ii municipal election at takoma park: The first e2e binding governmental election with ballot privacy," 2010.

[4] B. Adida, O. De Marneffe, O. Pereira, J.-J. Quisquater, *et al.*, "Electing a university president using open-audit voting: Analysis of real-world use of helios," *EVT/WOTE*, vol. 9, no. 10, 2009.

[5] B. Adida, "Helios: Web-based open-audit voting." 01 2008, pp. 335–348.

[6] P. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia, "PrÊt À voter: a voter-verifiable voting system," *Information Forensics and Security, IEEE Transactions on*, vol. 4, pp. 662 – 673, 01 2010.

[7] S. I. Mello, *A detailed forensic analysis and recommendations for Rhode Island's present and future voting systems*. University of Rhode Island, 2011.

[8] "Pilot implementation study of risk-limiting audit methods in the state of rhode island," https://verifiedvoting.org/wp-content/uploads/2020/07/RI-RLA-Report-2020.pdf, August 2019, (Accessed on 10/26/2020).

[9] "Es&s ds200 – verified voting," https://verifiedvoting.org/election-system/ess-ds200/, (Accessed on 10/27/2020).

[10] "Es&s ds850 & ds450 – verified voting," https://verifiedvoting.org/election-system/ess-ds850-ds450/, (Accessed on 10/27/2020).

[11] "Es&s automark – verified voting," https://verifiedvoting.org/election-system/ess-automark/, (Accessed on 10/27/2020).

[12] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," in *Conference on the Theory and Application of Cryptography*. Springer, 1990, pp. 437–455.

[13] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[14] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.

[15] "Blockchain explained: What is blockchain? — euromoney learning," https://www.euromoney.com/learning/blockchain-explained/what-is-blockchain, (Accessed on 10/28/2020).

[16] G. Governatori, F. Idelberger, Z. Milosevic, R. Riveret, G. Sartor, and X. Xu, "On legal contracts, imperative and declarative smart contracts, and blockchain systems," *Artificial Intelligence and Law*, vol. 26, no. 4, pp. 377–409, 2018.

[17] "Blockchain - wikipedia," https://en.wikipedia.org/wiki/Blockchain#cite_note-reason20160506-58, (Accessed on 10/28/2020).

[18] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Advances in Cryptology — CRYPTO' 92*, E. F. Brickell, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 139–147.

[19] M. Jakobsson and A. Juels, *Proofs of Work and Bread Pudding Protocols(Extended Abstract)*. Boston, MA: Springer US, 1999, pp. 258–272. [Online]. Available: https://doi.org/10.1007/978-0-387-35568-9_18

[20] R. K. Nichols, *ICSA guide to cryptography*. McGraw-Hill Professional, 1998.

[21] R. Rivest, "Rfc1321: The md5 message-digest algorithm," 1992.

[22] X. Wang, D. Feng, X. Lai, and H. Yu, "Collisions for hash functions md4, md5, haval-128 and ripemd." *IACR Cryptol. ePrint Arch.*, vol. 2004, p. 199, 2004.

[23] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full sha-1," in *Annual international cryptology conference*. Springer, 2005, pp. 17–36.

CHAPTER 3

A Blockchain-Based Voting Algorithm

A new voting algorithm that builds upon characteristics of blockchains is presented in this chapter. The procedure builds on other cryptographic techniques as well, including the El-Gamal public key encryption system and zero-knowledge protocols. This new algorithm offers advantages over current voting systems in terms of security and its ability to facilitate audits.

3.1 The Voting Process

On election day, each voter receives an unmarked paper ballot with no identifying index on it. This ballot follows the format now used in the state of Rhode Island. Hence there is no need to change or reconfigure the optical scanning devices currently used.

Voters mark their intentions on the ballot in the same manner currently used. After marking the ballot, they insert it into a ballot scanner. The scanners perform an initial validation to check for overvotes and other incorrectly marked ballots. Rhode Island elections currently uses ES&S DS200 optical scanners at polling places, and these machines have built-in features to detect incorrectly marked ballots and overvotes. If the scanner approves a ballot as correctly marked, it places the ballot into a bin attached to the scanner.

In our scheme, the scanner forwards the information on the ballot to a server located within the polling place to which it is "hard-wired". We refer to this server as the "validator", and its primary function is to store the information on all ballots cast in the precinct into a blockchain. After successfully saving the vote into the blockchain, the validator prints out a QR code on a printer attached to it. This QR code serves as the receipt for a particular ballot, and the voter can

take it home and use it to verify that his or her vote appears in the blockchain.

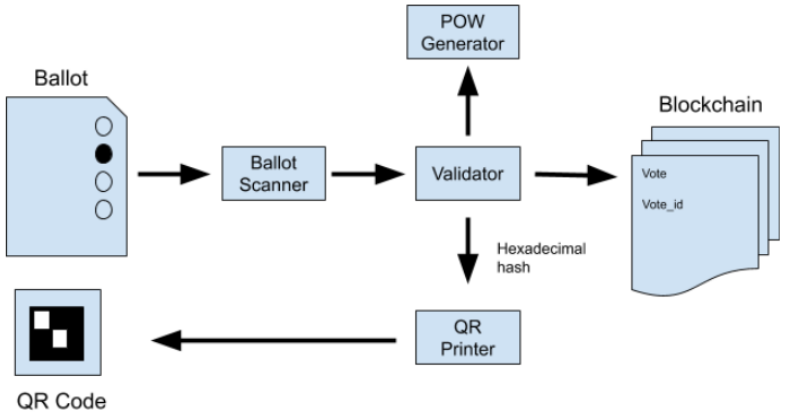

Figure 9. Voting process

3.2 Validator Authentication

Before storing a vote in the local chain, a validator performs a process called "proof of work" to generate the correct hash for the block representing the new ballot. Each hash generated by the validator needs to be formatted in a particular way. To achieve this format, a validator uses a random value called a nonce. This value is generated for each individual ballot by a secured workstation located in a centralized operation center. We refer to this workstation as the "central server". For security reasons validators do not store this random value in their memory and need to request it from the central server for each ballot.

To avoid unauthorized access to the central server, validators need to authenticate themselves to the central server. This authentication process is performed using a Fiat-Shamir zero-knowledge protocol [1]. The reason for using this protocol is to provide authentication without sharing any sensitive information between validators and the central server. This protects validators from network eavesdropping attacks.

The authentication process works as follows. Before the election, each valida-

tor and the central server generate private and public encryption/decryption keys using El-Gamal protocol [2]. This process is performed offline at a central election facility. The key distribution process is as follows:

1. First, both the central server and all of the validators agree on two values, a prime number n and a random value g. The central server and all validators use the same pair of values.

2. Next, the central server generates a random value a and computes $A = g^a \mod n$. The value a is the central server's "password" and A is its "username". The central server shares the value of A with all the validators and keeps a secret.

3. Each validator follows the same process, generating a random "password" b and "username" $B = g^b \mod n$. Every validator keeps its b value secret and sends its username to the central server, which stores the B values for each validator separately.

4. The central server uses the B value as the "username" for each validator in the Fiat-Shamir authentication process. It also computes $B^a \mod n$, which it uses as the public key to encrypt the proof of work it shares with a validator after successful authentication.

5. Each validator computes $A^b \mod n$ and uses it as the decryption key to decrypt the proof of work it receives from the central server after a successful authentication process.

On election day, each validator requests a new proof of work for every ballot. Before requesting a proof of work, a validator goes through a zero-knowledge check to authenticate itself to the central server. This process works as follows:

1. The validator selects a random value v and uses it to generate $T = g^v \mod n$, which it passes to the central server as its initial request for authentication.

2. The central server keeps the T value and responds to the request with a randomly generated value C.

3. The validator uses the equation $R = v - Cb$ based on the value C returned from the server and its secret password b. It forwards R as the second request in the authentication process.

4. The central server uses R, the previously generated value C and the validator's username B to generate $U = g^R B^C \mod n$.

5. If the value U is equal to the value T from the initial request, the central server accepts the proof of work request from the validator and establishes a connection to send the proof of work.

$$U = g^R B^C$$

$$U = g^{v-Cb}(g^b)^C$$

$$U = g^{v-Cb} g^{bC}$$

$$U = g^{v-Cb+Cb}$$

$$U = g^v = T$$

The information shared by the central server consists of two parts. The first is the proof of work, a random number (nonce) use to format the hash. The second is a timestamp bearing the time the central server generated the proof of work.

These two values are encrypted before sharing them with the validator. The central server uses the public key of the validator to encrypt the data. The validator uses its private key to decrypt the proof of work.

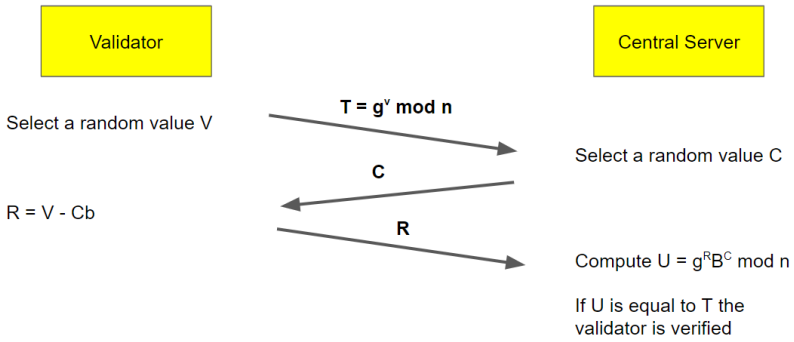

Figure 10. Validator authentication process

3.3 Proof of Work

Proof of work (POW) is the mechanism that protects the integrity of the information stored in the local chain. Blockchain technology provides an inherit resistance to alteration of data in the chain [3]. This resistance is achieved by storing the previous state of the chain, in the form of a hash value, in every block before the block is allowed to become a permanent part of the chain. The resistance is compromised, however, if someone tries to replace the entire blockchain with a new one. The concept of proof of work is used to protect against such attacks on the local chains stored in validators.

After authenticating a validator, the central server generates a random value (cryptographic nonce) and sends it back to the validator. The POW concept is based on this random number and "vote id", initially assigned a value of 0, that is stored in each block. The central server saves the newly generated nonce with the name of the validator that requested it and the time of the request in a POW log. This log is stored securely on the central server and is used to conduct post-election audits.

Since the POW generation algorithm is based on random numbers, a hacker with the same algorithm cannot predict the nonce generated by the central server in response to a specific request. The central server is the only place where the

POW (Nonce)	Timestamp	Validator (optional)
122345 106753	1589798902852 1589798902854	Validator_1 Validator_2

Figure 11. Proof of work log

nonce is stored in the entire system.

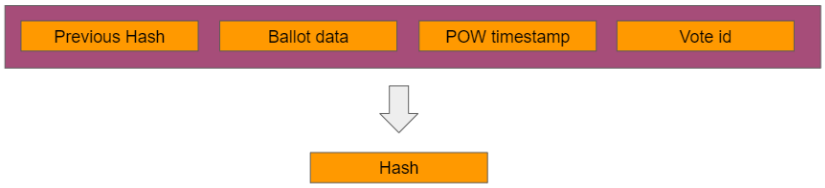

Figure 12. Block structure

Each block consists of four data fields as shown in Figure 12. The previous hash comes from the hash of the last block in the local chain, the ballot data is the information received from the ballot scanner, the POW timestamp is sent by the central server, and the vote id is the number used by the validator to generate and format the hash of the new block.

The POW is based on the validator generating a SHA256 has that begins with a binary sequence of 0s followed by a 1. See Figure 13. The number of 0s in the sequence determines the difficulty of the POW. The number of attempts required by the validator to find a hash with the correct format increases as the number of 0s increases. A feasible value for the number of 0s needs to be set before the start of the election. This value is static for all the blocks in every local chain during the election.

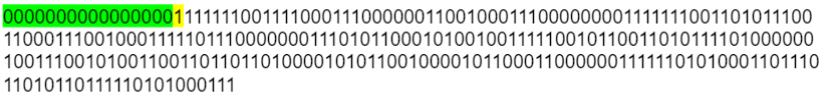

Figure 13. Hash format

To achieve the desired format, a validator adds the nonce value to the vote id. The validator generates a hash for the block combining all four data fields. If the hash is not in the expected format, the validator increases the vote id by one and generates another hash. The validator continues this process until it finds a hash of the correct format. When this hash is found, the validator generates a new block in the local chain. This block includes the data combined to generate the hash, but the vote id is altered by deducting the nonce. For example, if the server sends 152345 as the nonce, and the validator took 120 attempts to find a hash of the correct format, the vote id used to generate the correct hash is 152345 + 120 = 152465. But the validator stores 120 as the value for the vote id in the new block. As a result, validators retain no information about the nonce values used to generate each block.

After storing the block into the local chain, the validator uses the hash to generate a QR code that is printed on a piece of paper that voters can take home as a receipt of their vote. Voters can use this QR code to verify their vote has been included in the local chain. This voter verification process is referred to as a receipt audit.

3.4 Genesis Block

Recall that each block in the local chain stores the previous state of the chain in the form of a hash value. The problem is that each chain needs to start with a block that does not refer to a previous hash. This block is referred to as the "genesis block" of the chain. This special block is placed in each validator to mark the starting point of the local chain, and does not have any value for the "previous hash" and "ballot data" fields. Before election day, the central server generates a list of nonces and assigns each nonce to a genesis block in every validator. Each validator executes the POW procedure described above with null ballot data, a null

previous hash, and a timestamp for the nonce and to generate a "genesis block".

3.5 Voter Verification (Receipt Audit)

The main purpose of the voter verification process is to create an audit trail for each vote. Voters can also use it as evidence to verify their vote is included in the final tally.

After the conclusion of an election and the posting of the results, voters can access the vote verification system using the receipt they were given at the polls. Voters can use their smartphones or a QR code scanner to access the verification portal through an online website. After scanning the receipt, voters are directed to the verification portal for their precinct, where the system will confirm that their ballot was included in the tally. This confirmation does not reveal any detail about how the ballot was cast, just that the vote has been included in the election count.

3.6 Post-election Audits

The Rhode Island Board of Elections is now, under law, required to perform risk-limiting audits (RLAs) for certain elections [4]. Risk limiting audits can be of three principal types: ballot level comparison, ballot pooling, and batch level comparison. The system proposed here provides sufficient features to conduct all three types of RLAs.

3.6.1 Ballot Level Comparison

Since we have a separate chain for each precinct, this can be used to produce a separate final tally for each precinct. After the election, precincts can be selected randomly and the ballots for a precinct can be rescanned using a different scanner and validator to create a new chain. By generating the final tally for the new chain and comparing it to the original tally, we can evaluate the validity of the scanner

and validator used in the precinct. To improve the accuracy of the process, it can be repeated for different precincts.

Furthermore, we can manually evaluate a sample of ballots (e.g., 10%) from the precinct and project the election outcome for that precinct based on the tally of the sample. This is useful in verifying there are no programming issues related to ballot-scanning and processing the information on ballots.

3.6.2 Ballot Pooling

Sets of blocks from a precinct chain can be randomly selected to create a sample of ballots. By evaluating the information stored in each block, the vote count for the sample can be determined. This information can be used to estimate the final tally of the precinct, providing a projection of the election count. The same process can be performed manually by visually inspecting and counting the actual ballot papers (instead of using blocks in the chain) from a random sample of votes to provide an estimation of the final election tally.

3.6.3 Batch Level Comparison

This is one of the easiest audits to perform with our current design. Since the system provides the local blockchains for each precinct separately, they can be used as the batches. When the results for all precincts are added together, this should generate the final count for the election. We can select a random sample of precinct local chains and evaluate the tally of each chain to generate the final tally of the sample. This information can be used to estimate the final tally of the election. The count from the sample can be compared to the original tally from the logs generated by the voting machines.

3.6.4 Block Removal Method

In this procedure, a random set of blocks are removed from a precinct chain. The tally of the removed blocks and the rest of the blocks left in the chain is evaluated. The sum of the two tallies needs to be equal to the original final tally of the chain. The final tally can be validated with the equation in Figure 14.

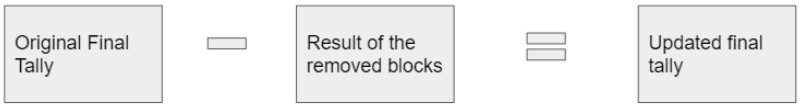

Figure 14. Block removal audit equation

This method provides mathematical proof of the tally in each precinct.

3.6.5 Block Connectivity Audit

In addition to the above audits, we can use the features of the blockchains to perform post-election audits. The validity of each link in the chain can be verified from the "previous_hash" value of the next block and the four pieces of information stored in the current block: the ballot data, timestamp, vote id, and the hash of the previous block. Verifying the validity of the links provides evidence that the local chain has not been maliciously altered. If the hash in either of the two consecutive blocks is incorrect, the connectivity check will fail.

3.6.6 Block Authenticity Audit

The proof of work for each block can also be used to validate the integrity of the chain. The timestamp stored in each block can be used to locate the corresponding POW (nonce) in the central server's log. After adding the nonce value to the vote_id value in the block, we can regenerate the hash of a block and check that the format of the hash is correct. By performing this test on all blocks in the chain, we can validate the integrity of the entire chain.

List of References

[1] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology — CRYPTO' 86*, A. M. Odlyzko, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 186–194.

[2] T. ELGAMAL, "A public key cryptosystem and a signature scheme based on discrete logarithms," https://caislab.kaist.ac.kr/lecture/2010/spring/cs548/basic/B02.pdf, (Accessed on 10/28/2020).

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[4] "Pilot implementation study of risk-limiting audit methods in the state of rhode island," https://verifiedvoting.org/wp-content/uploads/2020/07/RI-RLA-Report-2020.pdf, August 2019, (Accessed on 10/26/2020).

CHAPTER 4

Implementation and Testing

Implementation and testing of this research were conducted in two phases. In the first phase, a prototype was implemented based on the voting algorithm described in Chapter 3. In the second phase, three mock elections were conducted to test the system.

The first mock election tested the functionality of certain features in the system such as the zero-knowledge protocol, El-Gamal encryption, and proof of work. Only one precinct and a small number (less than 20) ballots were used in this mock election. No post-election audits were conducted due to an insufficient amount of data in the chain.

The second mock election analyzed the computational complexity of the proposed proof of work procedure to determine a feasible length for the sequence of 0s used in the proof of work. Only one precinct was used, and up to 100 votes tested for each length to collect data.

The third mock election replicated an actual election with two precincts. Up to 1000 ballots were used for each precinct. All the post-election audit methods were tested including voter verification. The findings of each mock election are presented at the end of this chapter.

4.1 Blockchain Voting Prototype

Actual ES&S DS200 optical scanners were not acquired for use in the prototype. Instead, the software was developed to simulate the functionality of a ballot scanner, a validator, and the central server. A webform developed using HTML and JavaScript was used for ballots. A web API (REST API) developed using the Python Flask framework was used to send the ballot data to the validator. Each

ballot was converted to a JSON object and stored inside the block as the value for the "ballot_data".

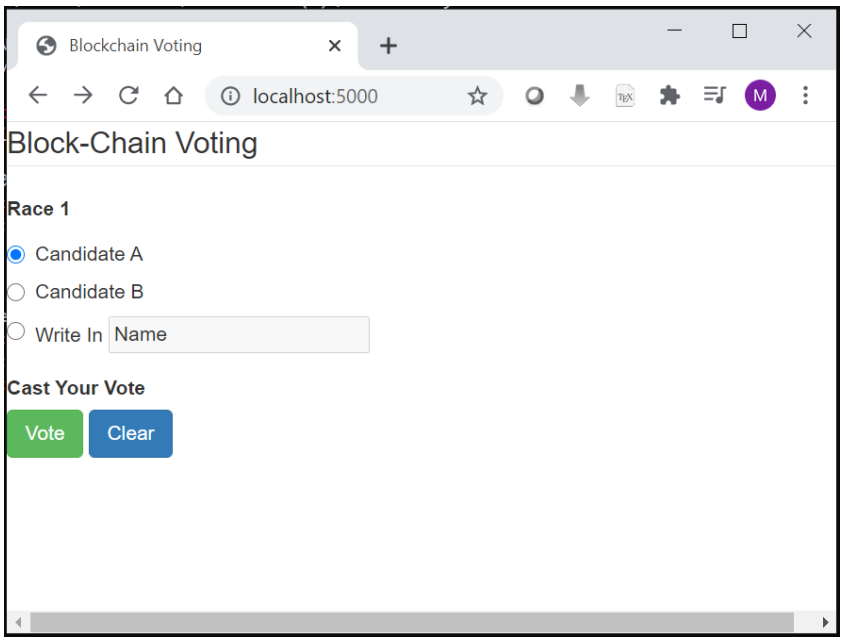

Figure 15. Online ballot form

The validator was implemented in Python, and the local chains were stored inside a validator as a binary file. All the algorithms (zero-knowledge protocol, El-Gamal encryption, and proof of work) used in the validator were implemented from scratch without using any third-party Python libraries. The built-in Python library "hashlib" was used to perform the SHA256 hash function.

Features of the central server were also implemented in python on the same server as the validator. The validator authentication process (zero-knowledge protocol) and the encryption of the data shared between the central server and the validator were implemented as described in Chapter 3. The proof of work log was also stored on the server as a binary file.

A separate tallying function was implemented to calculate the result of votes received by the validator. This function was written to replicate the tallying feature of an ES&S DS200 optical scanner. A separate web API was developed using the

Python Flask framework to view the final result of the election. The API returns a web page with the current tally of votes.

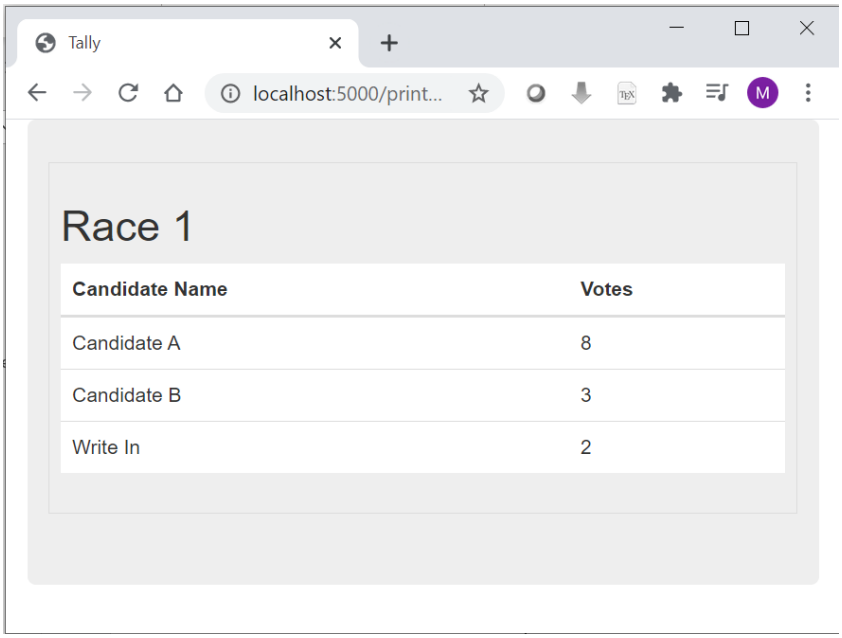

Figure 16. Tally API page

The voter verification process also faced minor changes from the original design due to the absence of actual validators attached to a QR printer. Instead of printing a receipt with the QR code, a QR code image was displayed on the online ballot page after a successfully cast vote. This image can be downloaded or printed on paper to use as a receipt to access the voter verification portal. As in the original design, this image can be scanned using a camera on a smartphone or a QR code scanner. The scanned image reveals a link to access the voter verification portal, which is a web API developed using the same Python Flask framework as the tally API. This API informs the voter that their vote is included in the final tally.

Post-elections audits were implemented as Python scripts. These scripts access the data stored in a binary file to perform the validations necessary to provide evidence of the accuracy of the election. The processes of voter verification and post-election audits are explained further in the following sections.

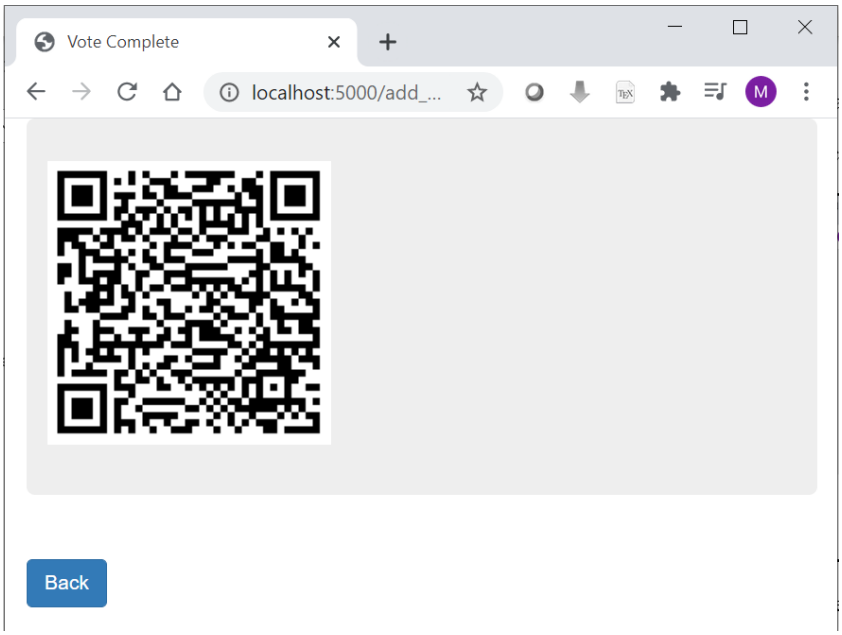

Figure 17. Vote verification QR code

4.2 First Mock Election

As previously mentioned, the main purpose of the first mock election was to test the key features of the voting algorithm. This mock election was designed to exercise functions such as generating public and private encryption keys for validators using El-Gamal key generation, authenticating validators to the central server with the Fiat-Shamir zero-knowledge protocol, and validating hashes using proof of work.

This mock election was conducted using a single precinct. Only one virtual validator was used to record votes. An online ballot form was designed to accommodate one race with three options to vote: two candidates plus a space for a write-in. Several votes were added to the chain representing all the options to check the success of the vote casting process.

All the keys and passwords required to perform the zero-knowledge protocol and data encryption were manually generated before the test and stored as fixed values inside the code.

4.2.1 El-Gamal Key Generation

For El-Gamal key generation, the central server and each of the validators need to agree upon two values: a prime number n and a random number g, where g is preferably a generator $\mod n$ (i.e., the powers of $g \mod n$ run through the numbers 1, 2, . . ., n–1 in some order). In the first mock test, 11881379 was the prime n and 1567892 was the value of g. These two values were saved inside both the central server and the validator. Then a random value $a = 15467$ was selected as the private key of the central server. By using the equation $A = g^a \mod n$, the public key for the validator was calculated and shared with all the validators.

$$A = g^a \mod n$$
$$A = 1567892^{15467} \mod 11881379$$
$$A = 11170411$$

The same process was repeated to generate private and public keys for the validator. Random value $b = 76543$ was selected as the private key for the validator and the public key was calculated and shared with the central server.

$$B = g^b \mod n$$
$$B = 1567892^{76543} \mod 11881379$$
$$B = 748829$$

This B value is also used as the validator's username for the Fiat-Shamir zero-knowledge protocol. The encryption key was generated in the central server using the equation $B^a \mod n$

$$Encryptkey = B^a \mod n$$

$$Encryptkey = 7488298^{15467} \mod 11881379$$

$$Encryptkey = 5956664$$

Similarly, the decryption key was generated for each validator using the equation $A^b \mod n$.

$$Decryptkey = A^b \mod n$$

$$Decryptkey = 11170411^{76543} \mod 11881379$$

$$Decryptkey = 5956664$$

4.2.2 Fiat-Shamir Zero-Knowledge Protocol

To test for successful validator authentication with the Fiat-Shamir zero-knowledge protocol, seven votes were added to the system using the fixed key values generated in the previous section for the validator. After adding the votes to the system, the local chain stored in the validator was retrieved to check whether all the votes were successfully recorded in the chain. The retrieved data showed the local chain held eight blocks, the genesis block plus one block for each of the seven votes cast. All the votes were successfully turned into blocks and stored in the chain (Figure 18). Hence the validator successfully authenticated itself to the central server for each vote cast without sharing any other information.

The system was also tested to be sure that an invalid authentication is detected and prevented. This was tested by using incorrect values for the username of the validator. To do so, the public key of the validator on the central server was changed to the incorrect value 5830. As with the previous test, several votes were added to the system to test the process. In this case, no votes were recorded in

[{'previous_hash': '', 'ballot_data': '', 'pow_timestamp': '', 'vote_id': 100}, {'previous_hash':
'c43ad61814a6cbb8bc6f498cec7788186b0acdfbbb11bbab88cfda9f3e363419', 'ballot_data': {'race': 'Candidate A'},
'pow_timestamp': 1601706835.7955282, 'vote_id': 76}, {'previous_hash':
'000ad7cbf1dd368d994f122e18f89fb278861e94a458d543ced6f291d16b8606', 'ballot_data': {'race': 'Candidate B'},
'pow_timestamp': 1601707422.5194328, 'vote_id': 472}, {'previous_hash':
'003ad25b15a66d6e1a38e996367e3983833393f75d65e2e87202281e5795d750', 'ballot_data': {'race': 'Candidate A'},
'pow_timestamp': 1601707976.4343178, 'vote_id': 0}, {'previous_hash':
'20caff9608604b144dd6bf26cd29378adf12cbb6ac25aa26a15218df388655f7', 'ballot_data': {'race': 'Candidate B'},
'pow_timestamp': 1601707982.8364346, 'vote_id': 6894}, {'previous_hash':
'd5e7b22496641abbf042380b764fc8bade0f3af639c0eeece585c5134d9904a8', 'ballot_data': {'race': 'Write In'},
'pow_timestamp': 1601707987.4587488, 'vote_id': 1081}, {'previous_hash':
'380a1da2e71443177bcaf777be6436894c9877006fe89c8daffa6ed7d21025b7', 'ballot_data': {'race': 'Candidate B'},
'pow_timestamp': 1601707992.765507, 'vote_id': 676}, {'previous_hash':
'64394580db11e34b19c6589f88da634c106b6d3a3707cd129239cfff1d98bef1', 'ballot_data': {'race': 'Candidate A'},
'pow_timestamp': 1601707997.3453271, 'vote_id': 150}]

Figure 18. Local ballot chain

the local chain and an error message was flagged in the program log to indicate the invalid login attempt. See Figure 19 and Figure 20.

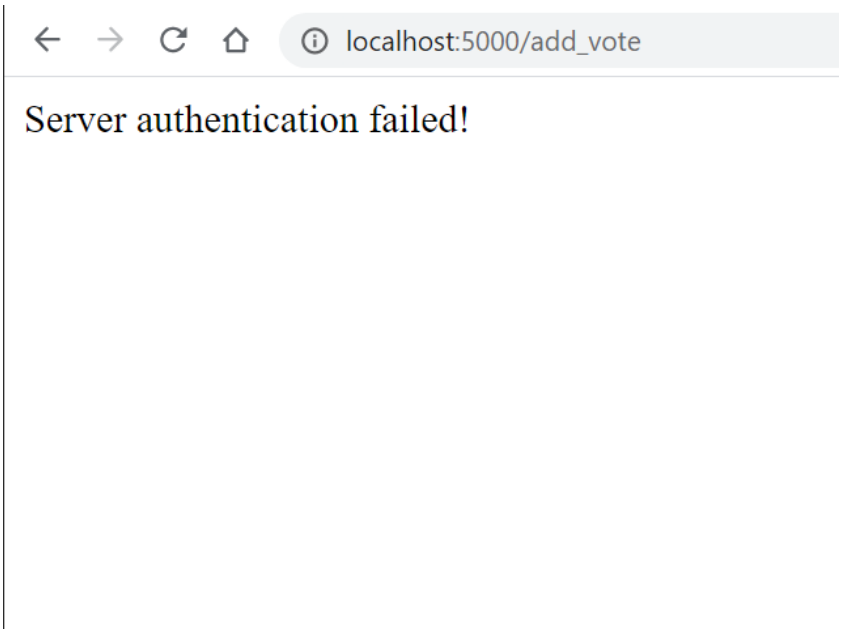

Figure 19. Server authentication error message

4.2.3 El-Gamal Data Encryption

A similar test was conducted to validate the El-Gamal data encryption process. First, several votes were added to the system. Then the nonce values in the pow_log file on the central server were checked against the program log to ensure the validator received the correct nonces for each vote. A fail scenario was

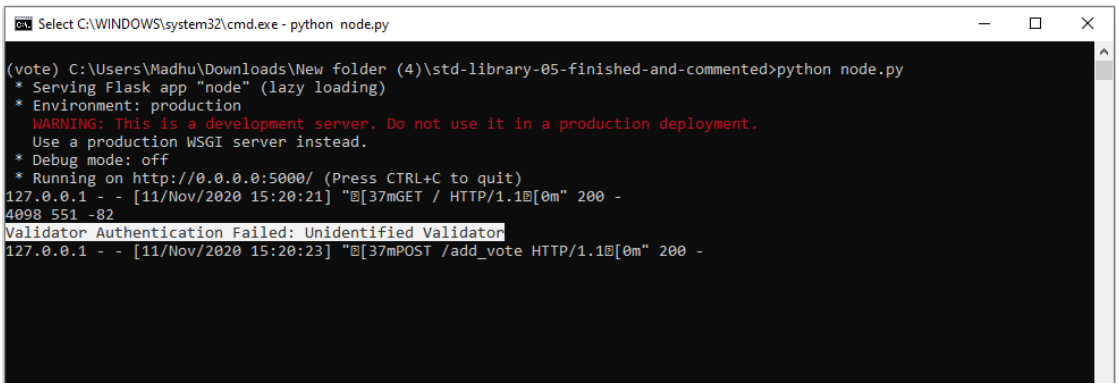

Figure 20. Server authentication error log

also tested by changing the private key of the validator to an incorrect value. A comparison between the generated hash and the corresponding pow for the block revealed the hashes generated with an invalid decryption key did not follow the correct pow rules.

The El-Gamal encryption test also provides evidence to demonstrate the success of the proof of work algorithm. All the hashes generated on the validator when the correct private key was used successfully followed the proof of work rule. The proof of work procedure is repeated until a validator finds a hash with the correct number of 0s. Figure 21 shows the last six hashes that did not match the pow rule and the final one which successfully matched the rule. The program accepts this last hash as a valid hash.

In addition to this information, the first mock election also revealed the success of storing and retrieving vote information and pow data in binary files. For the first mock election, a value of 16 was used as the length of the sequence of 0s needed to be followed by each hash. This number was selected to reduce the time required to generate the hash. Each successful hash was generated in an average time of 2020 milliseconds per nonce. The average number of guesses to generate a correct hash was 141212 tries.

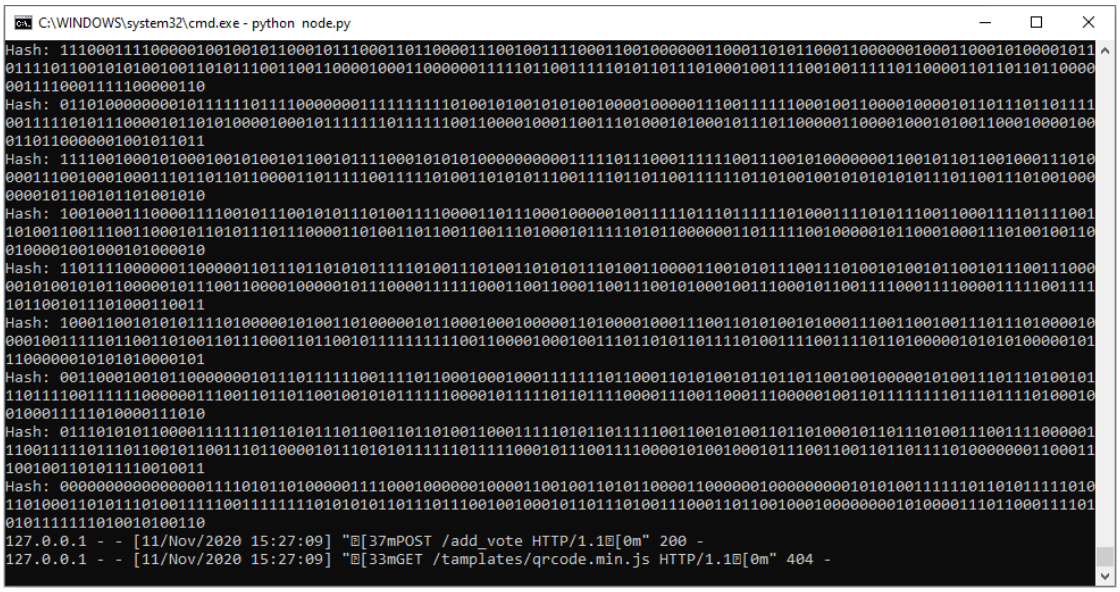

Figure 21. Hash generation process

4.3 Second Mock Election

The main purpose of the second mock election was to analyze the computational complexity of the proposed proof of work algorithm. The mock election was designed to determine the average number of attempts to find a correct hash as a function of the number of leading 0s. The test was conducted for seven different lengths starting from 8 to 20 in increments of two. For each length, 100 votes were added to the system using an API automation tool. The average number of attempts needed to format the hash was recorded along with the average time to generate a hash. The results of this test are shown in Table 1 and the graphs appearing in Figure 22 (average number of attempts) and Figure 23 (average time).

Based on these results, a value of 16 was selected as a feasible value for the number of 0s, and this value was used for the mock elections described in the next section.

POW (Length)	Average Tries	Average Time (ms)
8	516	560
10	2222	570
12	8464	630
14	28783	950
16	141212	2020
18	473874	5310
20	1791479	19050

Table 1. Proof of work performance analysis result

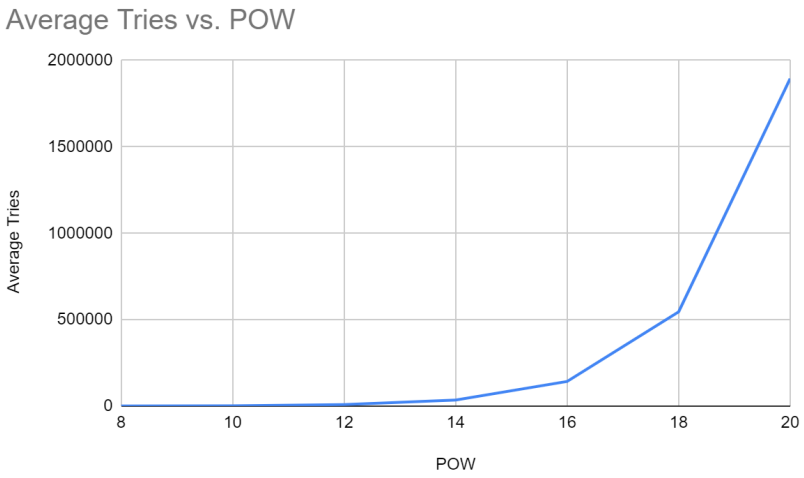

Figure 22. Average number of attempts vs length

4.4 Third Mock Election

The third mock election was conducted to simulate an actual election environment. Two precincts were used, each with its own validator and chain of votes. One reason for using two validators was to test the voter verification process and batch level comparison auditing. Approximately 1000 simulated ballots were cast for each precinct using an API automation tool. The same ballot structure was used as in the previous mock elections. The ballot consisted of one race with three options to vote: two candidates, and space for write-ins. Slightly biased ballot data was created for each precinct. All the votes were added to the system directly using the API instead of using the online ballot form.

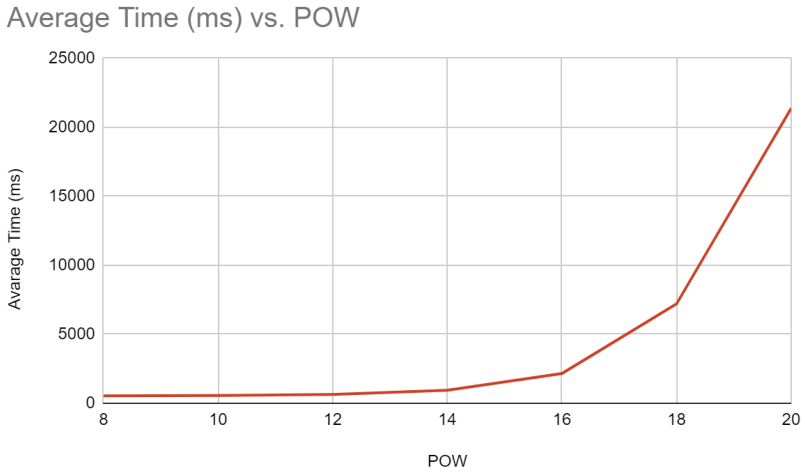

Figure 23. Average time vs length

Risk limiting audits and other validations were performed on each voting chain to analyze the system's capabilities to conduct post-election audits. The results of these audits and other artifacts of the election were cross-checked against the tally API to test the validity of the election results.

For the first precinct, 943 simulated votes were cast, 49% for Candidate A, 40% for Candidate B and the rest were write-ins. See Figure 25.

For the second precinct, 1213 simulated votes were cast, 30% for Candidate A, 46% for Candidate B and the rest were for write-ins. See Figure 26.

4.4.1 Ballot Level Comparison Audit

A ballot level comparison audit was implemented using a Python script. The script was written to access the binary file for the local chain from each precinct separately. A new tally was generated using the ballot data stored in each block. A comparison between the new tally and the tally generated by the voting machine was reported on a web page. This audit was conducted for both precincts and the results were identical to those for the votes cast in each case. See Figure 27 for the first precinct result and Figure 28 for the second precinct result.

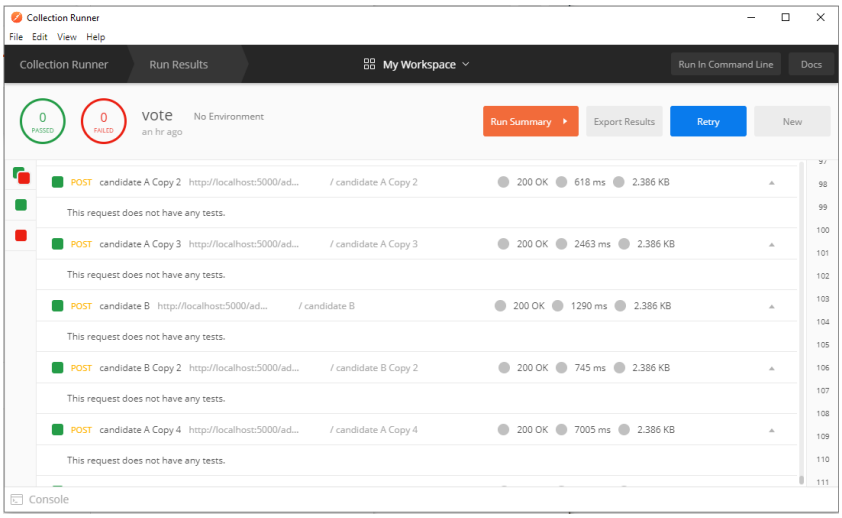

Figure 24. API automation tool (POSTMAN)

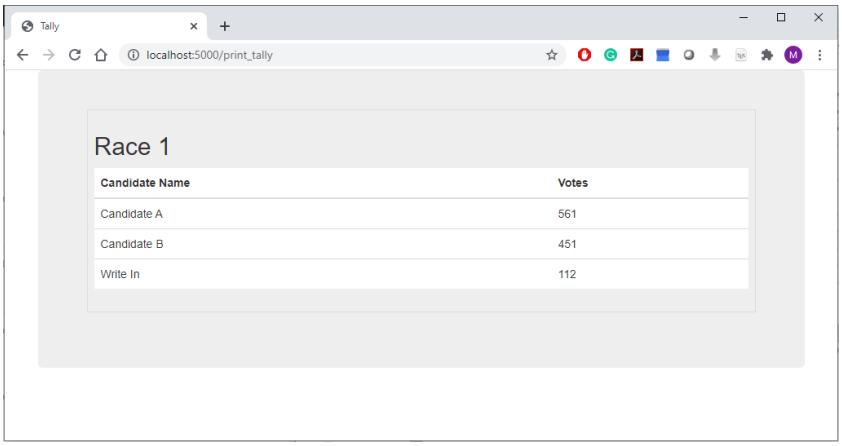

Figure 25. Tally of the first precinct

4.4.2 Ballot Pooling Audit

A ballot pooling audit was also conducted using a Python script. The script created a random sample of blocks from the precinct's local chain. For this test, 10% of the blocks in a local chain were selected as the sample size. The script produced the tally of the sample and used this to project the expected result for the entire chain. These values were reported on a web page along with the original machine tally for each precinct. The test was conducted on both precincts separately, and the result returned by the script provided evidence to support the

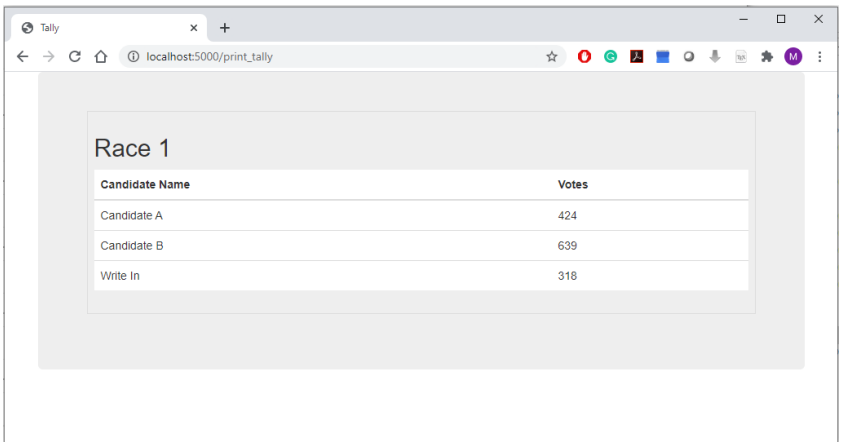

Figure 26. Tally of the second precinct

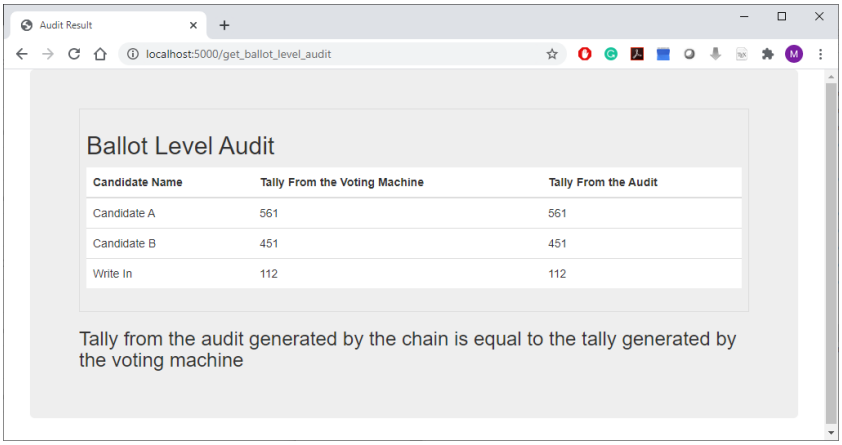

Figure 27. Ballot level comparison audit, first precinct

validity of votes produced by the tally API. The posted result for each precinct is shown in Figure 29 and 30.

4.4.3 Batch Level Comparison Audit

This audit was also conducted using a Python script. The tally for each precinct was produced separately using the script, and the sum of each precinct's tally was compared against the combined result of the tally API. The result was returned as a web page for comparison. The result is shown in Figure 31.

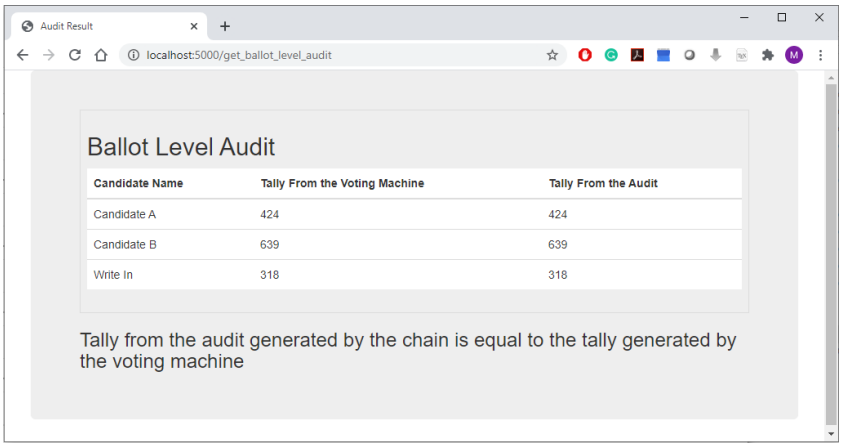

Figure 28. Ballot level comparison audit, second precinct

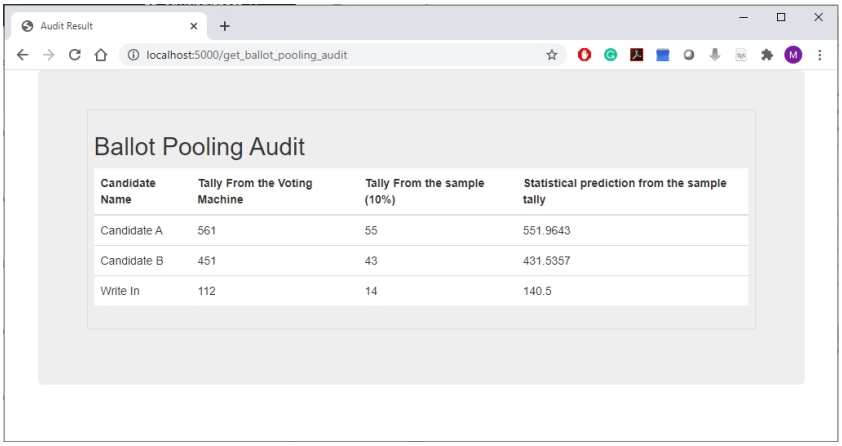

Figure 29. Ballot pooling audit, first precinct

4.4.4 Block Removal Audit

A block removal audit was conducted on the second precinct data using a Python script created for this purpose. First, a random sample of blocks was selected and these blocks were removed from the local chain. 10% of the total blocks in the chain were selected as the sample size. Both the tally of votes in the sample and the tally for the 90% of votes remaining in the local chain are calculated separately. The sum should agree with the votes cast on the voting machine. The results are shown in Figures 32 and 33.

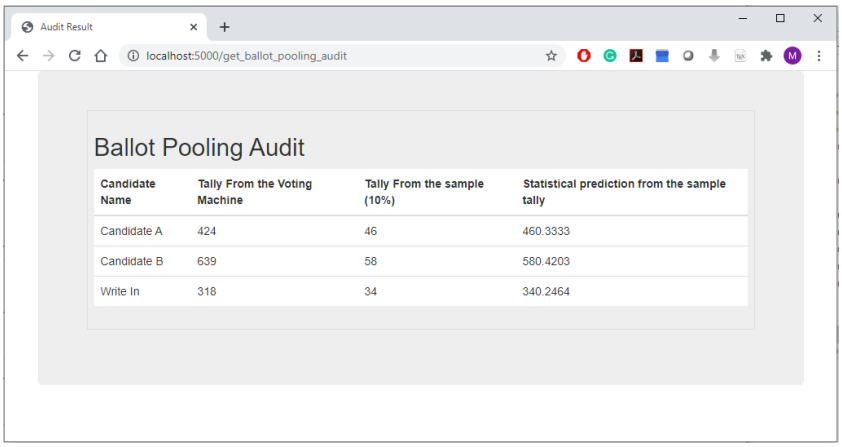

Figure 30. Ballot pooling audit, second precinct

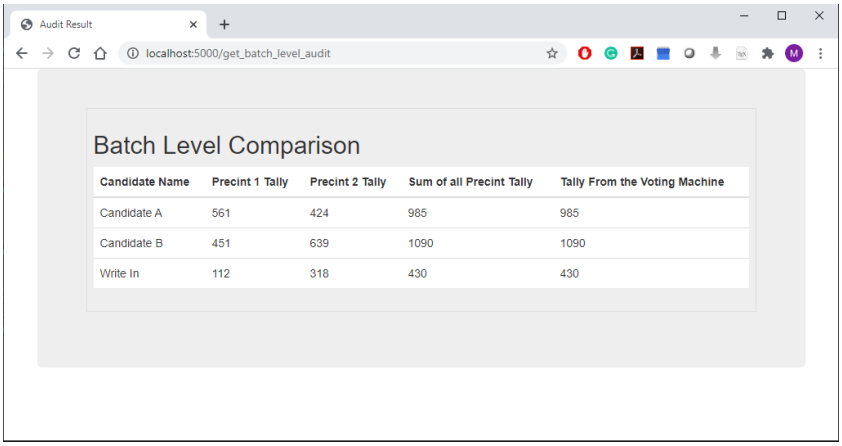

Figure 31. Batch level comparison audit

4.4.5 Block Authenticity Audit

This audit is conducted to check the validity of the data stored in the local chain. The audit is performed by recalculating the hash for each block and validating the newly generated hash against the original proof of work used for the block. Using the proof of work timestamp stored in each local block, the corresponding nonce value is extracted from the pow list on the central server. Then the hash of the block is re-calculated and checked against the proof of work. Each block in the chain is tested using a Python script that returns an error when a mismatch occurs. The script returns a list of the blocks in error for any mismatches. The

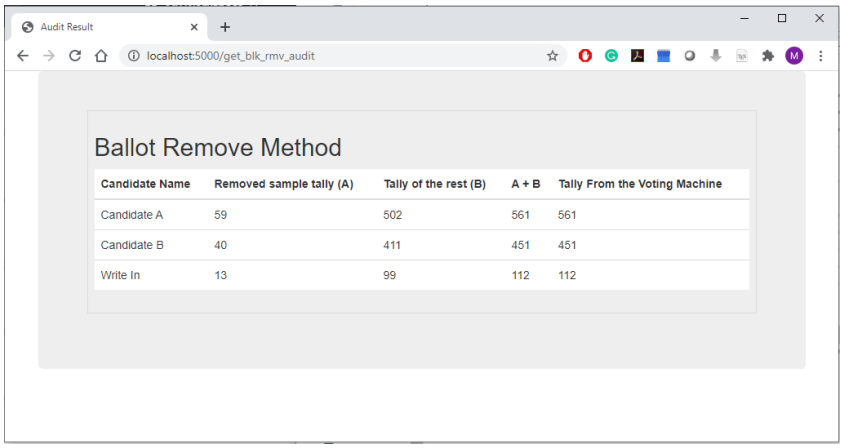

Figure 32. Block removal audit, first precinct

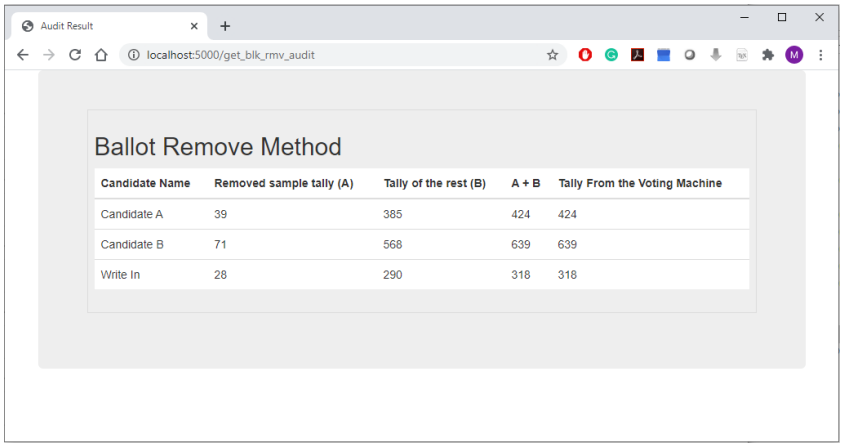

Figure 33. Block removal audit, second precinct

audit was conducted on the second precinct's data, and the successful result is shown in Figure 34.

To test a failure scenario, one hash value in the local chain that was validated in the previous test was altered. The result is shown in Figure 35.

4.4.6 Block Connectivity Audit

This audit is conducted to check the link between each block on the chain. An audit script was written to match the previous hash value in each block with the re-generated hash of the previous block. This test was applied to all of the blocks in the second precinct's chain. As with the block authenticity audit, both

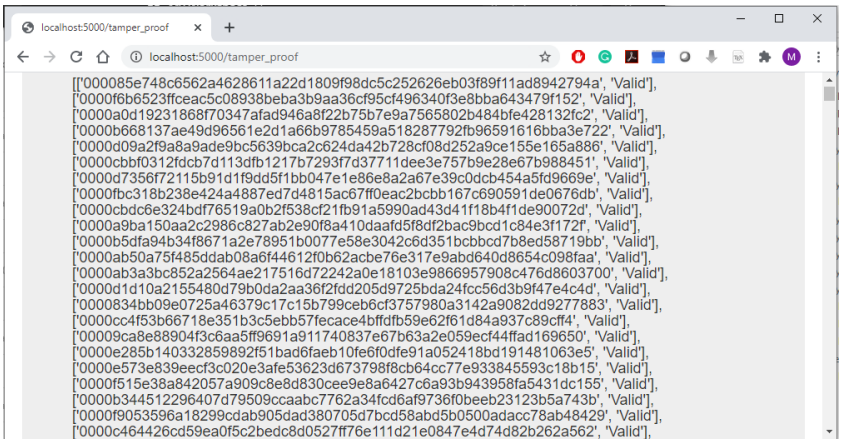

Figure 34. Block authenticity audit, success result

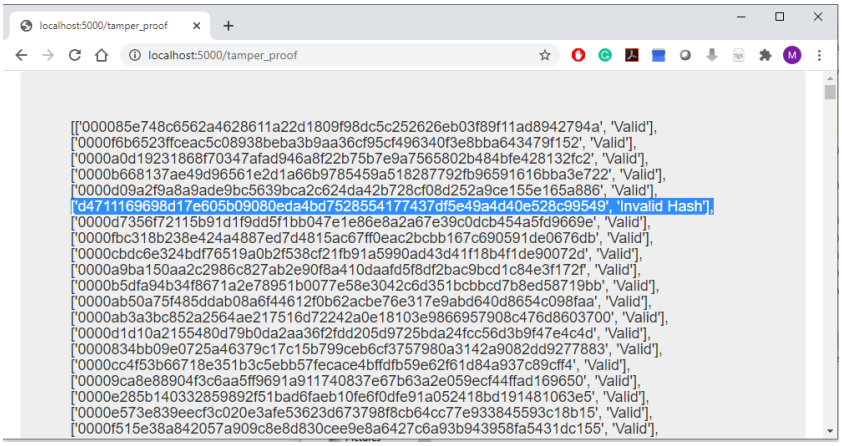

Figure 35. Block authenticity audit, failure result

success and failure scenarios were tested. To test a failure scenario, the hash value in a block was intentionally altered. The result of the successful scenario shown in Figure 36 and the failure scenario in Figure 37.

4.4.7 Voter Verification

Due to a change in the design of the QR code for vote verification, this code is not currently printed on a paper receipt as described in the third chapter. When a successful ballot is cast online, its QR code is instead displayed on the screen. This QR code follows the same structure as the printed QR code in the third chapter. See Figure 38.

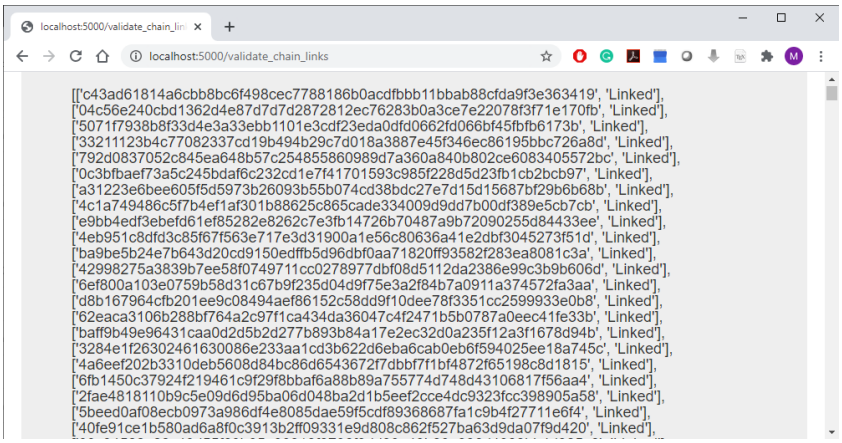

Figure 36. Block connectivity audit success

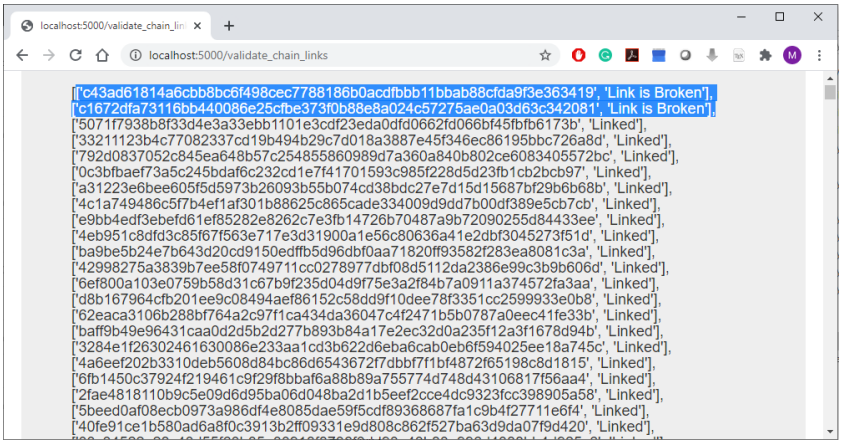

Figure 37. Block connectivity audit failure

Each QR code includes a link to the vote verification portal. The QR code also contains a hash of the four ballot data fields as a hexadecimal string.

To test the voter verification process, a successfully generated QR code was scanned using a mobile device. This device automatically logged into the voter verification portal through the link in the QR code. For valid votes, the system indicates the vote is successfully recorded (Figure 39), and for invalid votes returns an error (Figure 40).

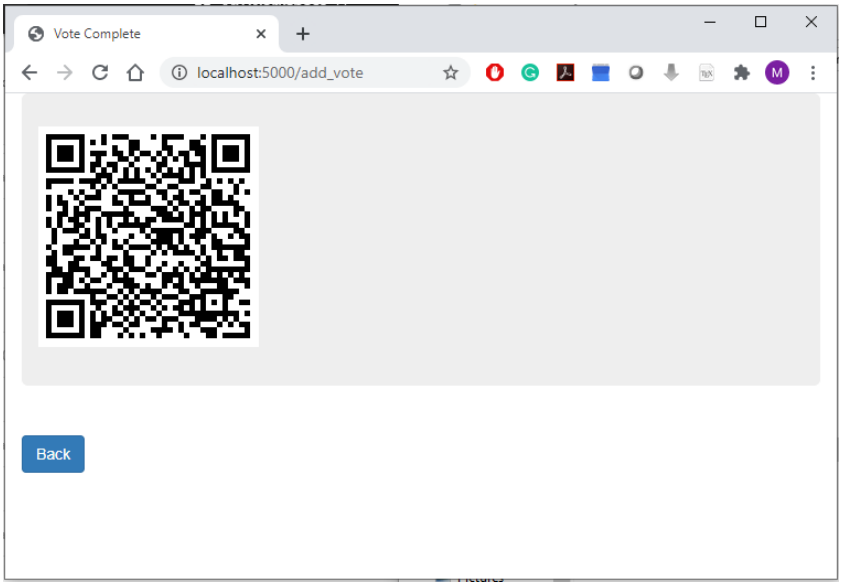

Figure 38. Vote verification QR page

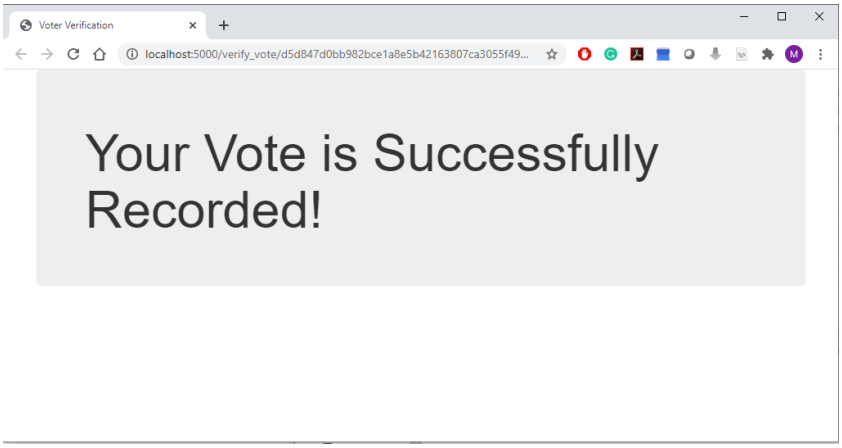

Figure 39. Vote verification success

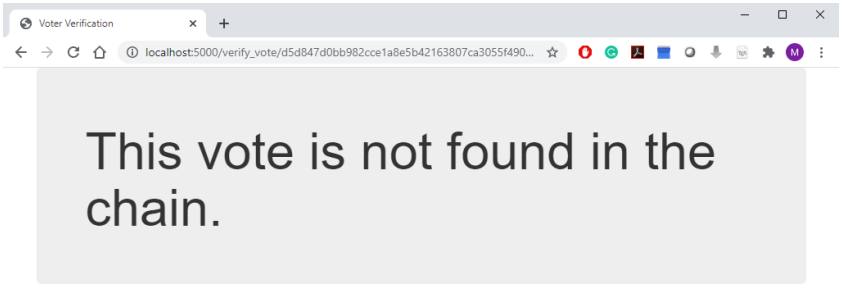

Figure 40. Vote verification failure

CHAPTER 5

Conclusion

The main objective of this research is to use features of blockchains and the notion of proof of work to create an auditable, immutable, and secure voting system. The system developed also incorporates, in a unique way, two additional security technologies. These are the use of zero-knowledge protocols to avoid sharing sensitive information, and end-to-end encrypted communication through the El-Gamal public-key cryptosystem.

The first mock election showed the Fiat Shamir zero-knowledge protocol to be a good way for a validator to authenticate itself without sharing any sensitive information. The procedure uses random numbers and calculations based on keys generated prior to the election for the authentication. This zero-knowledge protocol was tested more than 3000 times in the second and third mock elections. It successfully authenticated the validators to a central server on all occasions apart from a few times when the value R (from $R = v–Cb$) was negative. This issue was corrected by replacing g^R in the equation $U = g^R B^C \mod n$ with the modular inverse of g^{-R} for negative R values.

$$if(R > 0) : U = g^R B^C \mod n$$
$$else\_if(R < 0) : U = mod\_inverse(g^{-R}) B^C \mod n$$

An El-Gamal cryptosystem was used to generate secret keys for the validators and was also used to encrypt information shared between machines. This proved to be a successful method of achieving end-to-end encrypted communication between validators and the central server.

The performance of the proposed proof of work concept was tested and analyzed in the second mock election. Both the average number of attempts and the average time to compute a hash in the correct format was measured by increasing the length of the sequence of 0s in increments of two. The results revealed that the average number of attempts increased by approximately four times for every two 0s added to the sequence. The average time also showed an exponential increase for longer length sequences but not for shorter ones. This is mainly due to the time consumed performing the modular computations in the zero-knowledge protocol. Based on the results of these tests, a value of 16 was established as a good compromise for the number of 0s used in the rest of the study.

The final mock election was conducted to replicate an actual election environment. Risk-limiting audits, as required by Rhode Island law, were successfully conducted on more than 2000 ballots in two virtual precincts. In addition, three new post-election auditing methods were introduced and tested. These new audits take advantage of the characteristics of blockchains, and cannot be conducted with the election system currently in place. The resistance of the data in a block to alteration was tested with a block authenticity audit. The immutability of the chain was tested using a block connectivity audit. The paperless audit trail capability was tested using a block removal audit. The vote verification process was also tested during the third mock election.

The mock elections demonstrate that the characteristics of blockchains can be used to create an immutable and secure voting system. By storing the ballot data inside a blockchain-like data structure, any unauthorized modification to the data can be detected. The chain also creates an audit trail for each ballot that can be used in post-election audits. These experiments also demonstrated the proposed system's capability to perform several post-election audits including all

of the risk-limiting audits required by Rhode Island law [1].

One of the major parameters of this study was to introduce as few, if any, changes to the existing election process. The results of the second mock election indicate there will be no appreciable delay in the voting process. The time for the rest of the voting process remains unchanged.

Compared to many of the existing secure voting systems discussed in the second chapter, the system proposed here requires minimal effort to set up prior to election day. Moreover, no modification to the current ballot structure or voting machines used in Rhode Island is required.

Based on these observations, the proposed voting algorithm has clear potential to be integrated with current voting systems, thereby improving the security and integrity of elections.

5.1 Future Work

Several possible improvements were identified for both the prototype system and the underlying algorithm. These would be interesting to pursue as future work.

5.1.1 Improvements to the Prototype

An important improvement to the prototype would be to conduct mock elections with multiple validators working simultaneously to generate hashes. This would provide a better understanding of the performance of the system.

It would also be interesting to conduct an election allowing voters to vote for more than one candidate and to include more than one contest in the election. These features would create a much complex value for the "ballot data" field. The tally API would also need to be modified to accommodate these types of elections.

Finally, it would be interesting to test the prototype with an actual voting

machine, like the ES&S DS200, instead of an online ballot form to see how the prototype handles data provided by an actual voting machine.

5.1.2 Improvements to the Algorithm

One important improvement to the algorithm would be to use a single blockchain for all the validators in an election, instead of having separate local validators. This might require implementing a consensus mechanism similar to the bitcoin protocol [2] before adding a block to a validator's chain.

A second improvement would be the implementation of a private information retrieval protocol for the voter verification process [3]. This would create a secure retrieval of information from the blockchain when a voter tried to verify his or her vote. It would enhance privacy if no one knew the origin of voter verification requests.

List of References

[1] "Pilot implementation study of risk-limiting audit methods in the state of rhode island," https://verifiedvoting.org/wp-content/uploads/2020/07/RI-RLA-Report-2020.pdf, August 2019, (Accessed on 10/26/2020).

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[3] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proceedings of IEEE 36th Annual Foundations of Computer Science*. IEEE, 1995, pp. 41–50.

BIBLIOGRAPHY

"2000 united states presidential election in florida - wikipedia," https://en.wikipedia.org/wiki/2000_United_States_presidential_election_in_Florida, (Accessed on 10/26/2020).

"Blockchain - wikipedia," https://en.wikipedia.org/wiki/Blockchain#cite_note-reason20160506-58, (Accessed on 10/28/2020).

"Blockchain explained: What is blockchain? — euromoney learning," https://www.euromoney.com/learning/blockchain-explained/what-is-blockchain, (Accessed on 10/28/2020).

"Es&s automark – verified voting," https://verifiedvoting.org/election-system/ess-automark/, (Accessed on 10/27/2020).

"Es&s ds200 – verified voting," https://verifiedvoting.org/election-system/ess-ds200/, (Accessed on 10/27/2020).

"Es&s ds850 & ds450 – verified voting," https://verifiedvoting.org/election-system/ess-ds850-ds450/, (Accessed on 10/27/2020).

"List of close election results - wikipedia," https://en.wikipedia.org/wiki/List_of_close_election_results, (Accessed on 10/26/2020).

"Mediachain : Documentation," http://docs.mediachain.io/, (Accessed on 10/26/2020).

"Propy wiki," https://everipedia.org/wiki/lang_en/propy, (Accessed on 10/26/2020).

"State of rhode island general assembly. rhode island general laws, title17 - elections - chapter 17-19 conduct of election and voting equipment,and supplies," http://webserver.rilin.state.ri.us/Statutes/TITLE17/17-19/17-19-2.1.HTM, (Accessed on 10/26/2020).

"Pilot implementation study of risk-limiting audit methods in the state of rhode island," https://verifiedvoting.org/wp-content/uploads/2020/07/RI-RLA-Report-2020.pdf, August 2019, (Accessed on 10/26/2020).

Adida, B., "Helios: Web-based open-audit voting." 01 2008, pp. 335–348.

Adida, B., De Marneffe, O., Pereira, O., Quisquater, J.-J., *et al.*, "Electing a university president using open-audit voting: Analysis of real-world use of helios," *EVT/WOTE*, vol. 9, no. 10, 2009.

Carback, R., Chaum, D., Clark, J., Conway, J., Essex, A., Herrnson, P. S., Mayberry, T., Popoveniuc, S., Rivest, R. L., Shen, E., *et al.*, "Scantegrity ii municipal election at takoma park: The first e2e binding governmental election with ballot privacy," 2010.

Cathy Cox, J., "Georgia's unique model for election reform," https://votingmachines.procon.org/historical-timeline/, Nov 2002, (Accessed on 10/26/2020).

Chaum, D. *et al.*, "The scantegrity system, an introductory whitepaper and example," *Last accessed on January*, vol. 10, p. 2010, 2011.

Chor, B., Goldreich, O., Kushilevitz, E., and Sudan, M., "Private information retrieval," in *Proceedings of IEEE 36th Annual Foundations of Computer Science*. IEEE, 1995, pp. 41–50.

Cranor, L. F., *In Search of the Perfect Voting Technology: No Easy Answers*. Boston, MA: Springer US, 2003, pp. 17–30. [Online]. Available: https://doi.org/10.1007/978-1-4615-0239-5_2

Davenport, C., "Democrats blast ehrlich's absentee-voting initiative," https://votingmachines.procon.org/historical-timeline/, (Accessed on 10/26/2020).

Dwork, C. and Naor, M., "Pricing via processing or combatting junk mail," in *Advances in Cryptology — CRYPTO' 92*, Brickell, E. F., Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 139–147.

ELGAMAL, T., "A public key cryptosystem and a signature scheme based on discrete logarithms," https://caislab.kaist.ac.kr/lecture/2010/spring/cs548/basic/B02.pdf, (Accessed on 10/28/2020).

Essex, A., Clark, J., and Adams, C., "Aperio: High integrity elections for developing countries," in *Towards Trustworthy Elections*. Springer, 2010, pp. 388–401.

Fiat, A. and Shamir, A., "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology — CRYPTO' 86*, Odlyzko, A. M., Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 186–194.

FRIEDMAN, B., "Diebold voting machines can be hacked by remote control," https://www.salon.com/2011/09/27/votinghack/, (Accessed on 10/26/2020).

Ganji, R., "Electronic voting system using blockchain," https://education.dellemc.com/content/dam/dell-emc/documents/en-us/2018KS_Ganji-Electronic_Voting_System_using_Blockchain.pdf, (Accessed on 10/26/2020).

Governatori, G., Idelberger, F., Milosevic, Z., Riveret, R., Sartor, G., and Xu, X., "On legal contracts, imperative and declarative smart contracts, and blockchain systems," *Artificial Intelligence and Law*, vol. 26, no. 4, pp. 377–409, 2018.

Haber, S. and Stornetta, W. S., "How to time-stamp a digital document," in *Conference on the Theory and Application of Cryptography*. Springer, 1990, pp. 437–455.

Heller, D. A., "Certification of voter-verified paper audit trail printer completed," https://votingmachines.procon.org/historical-timeline/, July 2004, (Accessed on 10/26/2020).

Hjalmarsson, F. P., Hreioarsson, G. K., Hamdaqa, M., and Hjalmtysson, G., "Blockchain-based e-voting system," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. Los Alamitos, CA, USA: IEEE Computer Society, jul 2018, pp. 983–986. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CLOUD.2018.00151

Iansiti, M. and Lakhani, K., "The truth about blockchain:," *Harvard business review*, vol. 95, pp. 118–127, 01 2017.

Jakobsson, M. and Juels, A., *Proofs of Work and Bread Pudding Protocols(Extended Abstract)*. Boston, MA: Springer US, 1999, pp. 258–272. [Online]. Available: https://doi.org/10.1007/978-0-387-35568-9_18

Kevin Shelley, J., "California secretary of state news release," https://votingmachines.procon.org/historical-timeline/, (Accessed on 10/26/2020).

Mello, S. I., *A detailed forensic analysis and recommendations for Rhode Island's present and future voting systems*. University of Rhode Island, 2011.

Nakamoto, S., "Bitcoin: A peer-to-peer electronic cash system," *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

Narayanan, A., Bonneau, J., Felten, E., Miller, A., and Goldfeder, S., *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.

Nichols, R. K., *ICSA guide to cryptography*. McGraw-Hill Professional, 1998.

Park, S., Specter, M., Narula, N., and Rivest, R. L., "Going from bad to worse: from internet voting to blockchain voting."

Rivest, R., "Rfc1321: The md5 message-digest algorithm," 1992.

Ryan, P., Bismark, D., Heather, J., Schneider, S., and Xia, Z., "PrÊt À voter: a voter-verifiable voting system," *Information Forensics and Security, IEEE Transactions on*, vol. 4, pp. 662 – 673, 01 2010.

Songini, M. L., "Expert calls for increased e-voting security," https://www.computerworld.com/article/2560901/expert-calls-for-increased-e-voting-security.html, (Accessed on 10/26/2020).

Stark, P. B. and Wagner, D., "Evidence-based elections," *IEEE Security & Privacy*, vol. 10, no. 5, pp. 33–41, 2012.

Theisen, E., "Myth breakers: Facts about electronic elections," http://www.votersunite.org/mb2.pdf, (Accessed on 10/26/2020).

Wang, X., Feng, D., Lai, X., and Yu, H., "Collisions for hash functions md4, md5, haval-128 and ripemd." *IACR Cryptol. ePrint Arch.*, vol. 2004, p. 199, 2004.

Wang, X., Yin, Y. L., and Yu, H., "Finding collisions in the full sha-1," in *Annual international cryptology conference*. Springer, 2005, pp. 17–36.